



limHaloPT

one-loop halo model for LIM

Version 1.0.0

Developer: Azadeh Moradinezhad Dizgah

Generated by Doxygen 1.9.2

1 Summary	1
1.0.1 Dependencies	2
1.0.2 Compilation and Usage	2
1.0.3 Attribution	2
1.0.4 License	3
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	3
3.1 File List	3
4 Data Structure Documentation	4
4.1 Class_Cosmology_Struct Struct Reference	4
4.1.1 Detailed Description	4
4.1.2 Field Documentation	5
4.2 Cosmology Struct Reference	6
4.2.1 Detailed Description	6
4.2.2 Field Documentation	6
4.3 globals Struct Reference	7
4.3.1 Detailed Description	8
4.3.2 Field Documentation	8
4.4 integrand_parameters Struct Reference	11
4.4.1 Detailed Description	11
4.4.2 Field Documentation	12
4.5 integrand_parameters2 Struct Reference	13
4.5.1 Detailed Description	14
4.5.2 Field Documentation	14
4.6 Line Struct Reference	16
4.6.1 Detailed Description	16
4.6.2 Field Documentation	16
5 File Documentation	18
5.1 README.md File Reference	18
5.2 Global_Structs.h File Reference	18
5.3 Global_Structs.h	18
5.4 header.h File Reference	20
5.4.1 Macro Definition Documentation	22
5.4.2 Function Documentation	26
5.5 header.h	28
5.6 cosmology.c File Reference	30
5.6.1 Detailed Description	31
5.6.2 Function Documentation	32
5.6.3 Variable Documentation	61

5.7 IR_res.c File Reference	61
5.7.1 Detailed Description	62
5.7.2 Function Documentation	62
5.8 line_ingredients.c File Reference	76
5.8.1 Detailed Description	77
5.8.2 Function Documentation	78
5.8.3 Variable Documentation	108
5.9 main.c File Reference	108
5.9.1 Detailed Description	108
5.9.2 Function Documentation	108
5.9.3 Variable Documentation	111
5.10 ps_halo_1loop.c File Reference	111
5.10.1 Detailed Description	112
5.10.2 Function Documentation	113
5.10.3 Variable Documentation	126
5.11 ps_line_hm.c File Reference	126
5.11.1 Detailed Description	127
5.11.2 Function Documentation	127
5.11.3 Variable Documentation	143
5.12 ps_line_pt.c File Reference	143
5.12.1 Detailed Description	144
5.12.2 Function Documentation	144
5.12.3 Variable Documentation	150
5.13 survey_specs.c File Reference	150
5.13.1 Detailed Description	151
5.13.2 Function Documentation	151
5.13.3 Variable Documentation	155
5.14 utilities.c File Reference	156
5.14.1 Detailed Description	156
5.14.2 Function Documentation	157
5.15 wnw_split.c File Reference	162
5.15.1 Detailed Description	163
5.15.2 Function Documentation	163

1 Summary

Author: Azadeh Moradinezhad Dizgah

with contributions from Farnik Nikakhtar

Welcome to limHaloPT, a numerical package for computing the clustering and shot-noise contributions to the power spectrum of line intensity/temperature fluctuations within halo-model framework. The current version of the code, is limited to real-space, and redshift-space distortions will be included in the next release.

The extended halo model of line intensity power spectrum implemented in limHaloPT, combines the predictions of EFTofLSS for halo power spectrum with the standard halo model to account for the nonlinear evolution of matter fluctuations and the nonlinear biasing relation between line intensity fluctuations and the underlying dark matter distribution in 2-halo term. Furthermore, the model includes the effect of large bulk velocities (Infrared Resummation) in the 2-halo term. The deviations from Poisson shot noise on large scales are also computed within the halo model.

This package is released together with the following publication, [arXiv:2111.03717](#), where the prediction of the model are tested against new suite of simulated intensity (brightness temperature) maps of CO and [CII] lines. The meshed files from MithraLIMSims will be publicly available on [MithraLIMSims](#). The code to analyse the simulated maps is an extension of the toolkit used in analysis Hidden-Valley simulations, and is publicly available on [LIM Analysis](#). As discussed in the manuscript above, the packages to compute the theory predictions and creating the simulated intensity maps can be straightforwardly extended to compute the power spectrum signal of other emission lines (emitted from star-forming galaxies), beside CO and [CII].

The source code of this package is publicly available on GitHub at [limHaloPT](#).

1.0.1 Dependencies

The limHaloPT package calls various functions from [CLASS](#) Boltzman solver, including the matter power spectrum and transfer functions, growth factor etc. Therefore, you need to first download and compile CLASS code, and place the "libclass.a" file in the "CLASS/lib/" folder. Furthermore, the loop calculations are performed with direct numerical integration, using routines of [CUBA](#) library. Furthermore, the code heavily uses functions of [GSL](#) scientific library. Therefore, make sure that the two libraries are correctly linked to limHaloPT by making necessary modifications to the makefile (placed in Source directory) of limHaloPT package.

1.0.2 Compilation and Usage

- To compile, type: make
- To run, type: ./limHaloPT

If you modified the code, you need to first do "make clean" before doing "make". Depending on what quantities you want to calculate, you can modify the `main()` function in `main.c` module (as marked in the code). As examples, I have included the calls to two functions to compute the clustering and shot noise contributions.

1.0.3 Attribution

You can use this package freely, provided that in your publication you cite the following paper: Moradinezhad, Nikakhtar, Keating, Castorina: [arXiv:2111.03717](#). Furthermore, since limHaloPT relies on CLASS Boltzman code, you should also cite at least this paper [arxiv:1104.2933](#) as required by CLASS developers.

1.0.4 License

Copyright 2021 Azadeh Moradinezhad Dizgah.

limHaloPT is free software made available under the MIT License. For details see the [LICENSE](#) file.

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Class_Cosmology_Struct	
Structure to store cosmology structure from CLASS code	4
Cosmology	
Structure that holds varioud quantities that need to be evaluated for a given choice of cosmological paramteres	6
globals	
A global structure including the values of cosmological parmaeters, 2d interpolator of SFR, and names of various files	7
integrand_parameters	
A structure passed to the integrators to hold the parameters fixed in the integration	11
integrand_parameters2	
Another structure passed to the integrators to hold the parameters fixed in the integration	13
Line	
Structure that holds the Line-related quantities, including the interpolators for first and second moments of the line luminosity and the linear and quadratic luminosity-weighted line biases	16

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

Global_Structs.h	18
header.h	20
cosmology.c	
Documented cosmology module	30
IR_res.c	
Documented IR_res module	61
line_ingredients.c	
Documented line_ingredients module	76

main.c

Documented main module, including functions to initialize and cleanup the cosmology structure and examples of calls to functions in other modules to compute the line clustering and shot power spectrum

108

ps_halo_1loop.c

Documented real-space, direct integration computation of 1loop contributions of the halo/galaxy power spectrum See arXiv:2010.14523 for explicit expressions

111

ps_line_hm.c

Documented halo-model computation of line power spectrum, including clustering and stochastic contributions beyond Poisson limit

126

ps_line_pt.c

Documented computation of Poisson shot noise and tree-level line power spectrum in real and redshift-space

143

survey_specs.c

Documented computation of some survey-related functions

150

utilities.c

Documented basic utility functions used by other modules of the code

156

wnw_split.c

Documented wiggle-nowiggle split based on 3d Gaussian filter in linear k, and using the Eisenstein-Hu wiggle-no wiggle template

162

4 Data Structure Documentation

4.1 Class_Cosmology_Struct Struct Reference

Structure to store cosmology structure from CLASS code.

```
#include <Global_Structs.h>
```

Data Fields

- struct precision [pr](#)
- struct background [ba](#)
- struct thermo [th](#)
- struct perturbs [pt](#)
- struct transfers [tr](#)
- struct primordial [pm](#)
- struct spectra [sp](#)
- struct nonlinear [nl](#)
- struct lensing [le](#)
- struct output [op](#)
- ErrorMessage [errmsg](#)

4.1.1 Detailed Description

Structure to store cosmology structure from CLASS code.

4.1.2 Field Documentation

4.1.2.1 **ba** struct background ba

4.1.2.2 **errmsg** ErrorMsg errmsg

4.1.2.3 **le** struct lensing le

4.1.2.4 **nl** struct nonlinear nl

4.1.2.5 **op** struct output op

4.1.2.6 **pm** struct primordial pm

4.1.2.7 **pr** struct precision pr

4.1.2.8 **pt** struct perturbs pt

4.1.2.9 **sp** struct spectra sp

4.1.2.10 **th** struct thermo th

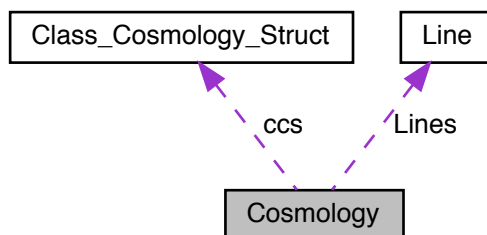
4.1.2.11 **tr** struct transfers tr

4.2 Cosmology Struct Reference

Structure that holds varioud quantities that need to be evaluated for a given choice of cosmological paramteres.

```
#include <Global_Structs.h>
```

Collaboration diagram for Cosmology:



Data Fields

- struct [Class_Cosmology_Struct](#) [ccs](#)
- struct [Line](#) ** [Lines](#)
- int [NLines](#)
- long [mode_nu](#)
- double [cosmo_pars](#) [6]

4.2.1 Detailed Description

Structure that holds varioud quantities that need to be evaluated for a given choice of cosmological paramteres.

This includes, the [Class_Cosmology_Struct](#) (initialized in [cosmology.c](#)), and [Line](#) Structure (initialized in [line_ingredients.c](#)).

4.2.2 Field Documentation

4.2.2.1 ccs struct [Class_Cosmology_Struct](#) [ccs](#)

4.2.2.2 cosmo_pars double [cosmo_pars](#)[6]

4.2.2.3 Lines `struct Line** Lines`

4.2.2.4 mode_nu `long mode_nu`

4.2.2.5 NLines `int NLines`

4.3 globals Struct Reference

A global structure including the values of cosmological parameters, 2d interpolator of SFR, and names of various files.

```
#include <Global_Structs.h>
```

Data Fields

- double `H0`
- double `c`
- double `As`
- double `logAs`
- double `ns`
- double `h`
- double `Omega_cdm`
- double `Omega_b`
- double `Omega_r`
- double `Omega_lambda`
- double `Omega_g`
- double `Omega_nu`
- double `b1`
- double `sigFOG0`
- long `Npars`
- double `z_i`
- double `rho`
- double `mass`
- double `kp`
- double `ng`
- double `volume`
- double `kf`
- double `h_m`
- double `M_min`
- double `M_max`
- double `z_max`
- char `project_home` [FILENAME_MAX]
- char `output_dir` [FILENAME_MAX]
- char `data_dir` [FILENAME_MAX]
- char `data_priors` [FILENAME_MAX]
- double `PS_kmin`
- double `PS_kmax`
- char `SFR_filename` [FILENAME_MAX]
- char `Planck_Fisher_filename` [FILENAME_MAX]
- gsl_interp_accel * `logM_accel_ptr`
- gsl_interp_accel * `z_accel_ptr`
- gsl_spline2d * `logSFR_spline2d_ptr`

4.3.1 Detailed Description

A global structure including the values of cosmological parameters, 2d interpolator of SFR, and names of various files.

4.3.2 Field Documentation

4.3.2.1 **As** `double As`

4.3.2.2 **b1** `double b1`

4.3.2.3 **c** `double c`

4.3.2.4 **data_dir** `char data_dir[FILENAME_MAX]`

4.3.2.5 **data_priors** `char data_priors[FILENAME_MAX]`

4.3.2.6 **h** `double h`

4.3.2.7 **H0** `double H0`

4.3.2.8 **h_m** `double h_m`

4.3.2.9 **kf** `double kf`

4.3.2.10 **kp** double kp

4.3.2.11 **logAs** double logAs

4.3.2.12 **logM_accel_ptr** gsl_interp_accel* logM_accel_ptr

4.3.2.13 **logSFR_spline2d_ptr** gsl_spline2d* logSFR_spline2d_ptr

4.3.2.14 **M_max** double M_max

4.3.2.15 **M_min** double M_min

4.3.2.16 **mass** double mass

4.3.2.17 **ng** double ng

4.3.2.18 **Npars** long Npars

4.3.2.19 **ns** double ns

4.3.2.20 **Omega_b** double Omega_b

4.3.2.21 Omega_cdm double Omega_cdm

4.3.2.22 Omega_g double Omega_g

4.3.2.23 Omega_lambda double Omega_lambda

4.3.2.24 Omega_nu double Omega_nu

4.3.2.25 Omega_r double Omega_r

4.3.2.26 output_dir char output_dir[FILENAME_MAX]

4.3.2.27 Planck_Fisher_filename char Planck_Fisher_filename[FILENAME_MAX]

4.3.2.28 project_home char project_home[FILENAME_MAX]

4.3.2.29 PS_kmax double PS_kmax

4.3.2.30 PS_kmin double PS_kmin

4.3.2.31 rho double rho

4.3.2.32 SFR_filename `char SFR_filename[FILENAME_MAX]`

4.3.2.33 sigFOG0 `double sigFOG0`

4.3.2.34 volume `double volume`

4.3.2.35 z_accel_ptr `gsl_interp_accel* z_accel_ptr`

4.3.2.36 z_i `double z_i`

4.3.2.37 z_max `double z_max`

4.4 integrand_parameters Struct Reference

A structure passed to the integrators to hold the parameters fixed in the integration.

```
#include <header.h>
```

Data Fields

- double [p1](#)
- double [p2](#)
- double [p3](#)
- double [p4](#)
- double [p5](#)
- double [p6](#)
- double [p7](#)
- double [p8](#)
- double [p9](#)
- double [p10](#)
- double [p11](#)
- long [p12](#)
- long [p13](#)

4.4.1 Detailed Description

A structure passed to the integrators to hold the parameters fixed in the integration.

4.4.2 Field Documentation

4.4.2.1 p1 double p1

4.4.2.2 p10 double p10

4.4.2.3 p11 double p11

4.4.2.4 p12 long p12

4.4.2.5 p13 long p13

4.4.2.6 p2 double p2

4.4.2.7 p3 double p3

4.4.2.8 p4 double p4

4.4.2.9 p5 double p5

4.4.2.10 p6 double p6

4.4.2.11 p7 double p7

4.4.2.12 p8 double p8

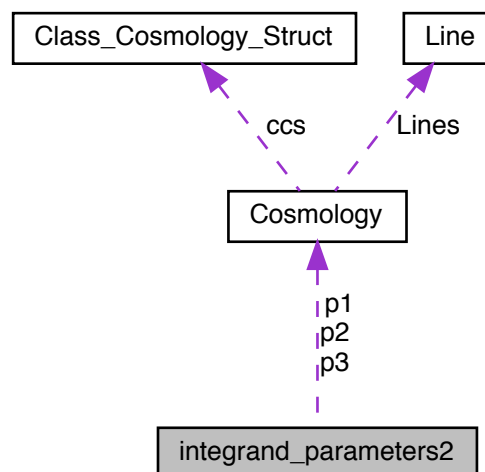
4.4.2.13 p9 double p9

4.5 integrand_parameters2 Struct Reference

Another structure passed to the integrators to hold the parameters fixed in the integration.

```
#include <header.h>
```

Collaboration diagram for integrand_parameters2:



Data Fields

- struct [Cosmology](#) * p1
- struct [Cosmology](#) * p2
- struct [Cosmology](#) * p3
- double p4
- double p5
- double p6
- double p7
- double p8
- double p9

- double [p10](#)
- double [p11](#)
- double [p12](#)
- long [p13](#)
- long [p14](#)
- long [p15](#)
- long [p16](#)
- long [p17](#)
- long [p18](#)
- int [p19](#)
- double * [p20](#)
- size_t [p22](#)

4.5.1 Detailed Description

Another structure passed to the integrators to hold the parameters fixed in the integration.

4.5.2 Field Documentation

4.5.2.1 p1 struct [Cosmology](#)* p1

4.5.2.2 p10 double p10

4.5.2.3 p11 double p11

4.5.2.4 p12 double p12

4.5.2.5 p13 long p13

4.5.2.6 p14 long p14

4.5.2.7 p15 long p15

4.5.2.8 p16 long p16

4.5.2.9 p17 long p17

4.5.2.10 p18 long p18

4.5.2.11 p19 int p19

4.5.2.12 p2 struct [Cosmology](#)* p2

4.5.2.13 p20 double* p20

4.5.2.14 p22 size_t p22

4.5.2.15 p3 struct [Cosmology](#)* p3

4.5.2.16 p4 double p4

4.5.2.17 p5 double p5

4.5.2.18 **p6** double p6

4.5.2.19 **p7** double p7

4.5.2.20 **p8** double p8

4.5.2.21 **p9** double p9

4.6 Line Struct Reference

Structure that holds the Line-related quantities, including the interpolators for first and second moments of the line luminosity and the linear and quadratic luminosity-weighted line biases.

```
#include <Global_Structs.h>
```

Data Fields

- long [LineType](#)
- int [initialized](#)
- size_t [npointsInterp](#)
- double [line_freq](#)
- gsl_interp_accel * [mom1_accel_ptr](#)
- gsl_spline * [mom1_spline_ptr](#)
- gsl_interp_accel * [mom2_accel_ptr](#)
- gsl_spline * [mom2_spline_ptr](#)
- gsl_interp_accel * [b1_LW_accel_ptr](#)
- gsl_spline * [b1_LW_spline_ptr](#)
- gsl_interp_accel * [b2_LW_accel_ptr](#)
- gsl_spline * [b2_LW_spline_ptr](#)

4.6.1 Detailed Description

Structure that holds the Line-related quantities, including the interpolators for first and second moments of the line luminosity and the linear and quadratic luminosity-weighted line biases.

4.6.2 Field Documentation

4.6.2.1 b1_LW_accel_ptr `gsl_interp_accel* b1_LW_accel_ptr`

4.6.2.2 b1_LW_spline_ptr `gsl_spline* b1_LW_spline_ptr`

4.6.2.3 b2_LW_accel_ptr `gsl_interp_accel* b2_LW_accel_ptr`

4.6.2.4 b2_LW_spline_ptr `gsl_spline* b2_LW_spline_ptr`

4.6.2.5 initialized `int initialized`

4.6.2.6 line_freq `double line_freq`

4.6.2.7 LineType `long LineType`

4.6.2.8 mom1_accel_ptr `gsl_interp_accel* mom1_accel_ptr`

4.6.2.9 mom1_spline_ptr `gsl_spline* mom1_spline_ptr`

4.6.2.10 mom2_accel_ptr `gsl_interp_accel* mom2_accel_ptr`

4.6.2.11 mom2_spline_ptr `gsl_spline* mom2_spline_ptr`

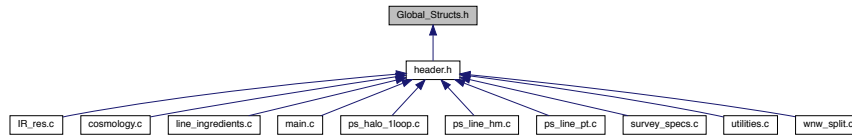
4.6.2.12 npointsInterp `size_t npointsInterp`

5 File Documentation

5.1 README.md File Reference

5.2 Global_Structs.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [Class_Cosmology_Struct](#)
Structure to store cosmology structure from CLASS code.
- struct [Cosmology](#)
Structure that holds varioud quantities that need to be evaluated for a given choice of cosmological paramteres.
- struct [Line](#)
Structure that holds the Line-related quantities, including the interpolators for first and second moments of the line luminosity and the linear and quadratic luminosity-weighted line biases.
- struct [globals](#)
A global structure including the values of cosmological parmaeters, 2d interpolator of SFR, and names of various files.

5.3 Global_Structs.h

[Go to the documentation of this file.](#)

```

1
2
3 #ifndef GLOBALSTRUCTS_H_
4 #define GLOBALSTRUCTS_H_
5
6
7
8
9 struct Class_Cosmology_Struct{
10
11     struct precision          pr;          /* for precision parameters */
12     struct background         ba;          /* for cosmological background */
13     struct thermo             th;          /* for thermodynamics */
14     struct perturb            pt;          /* for source functions */
15     struct transfers           tr;          /* for transfer functions */
16     struct primordial          pm;          /* for primordial spectra */
17     struct spectra             sp;          /* for output spectra */
18     struct nonlinear           nl;          /* for non-linear spectra */
19     struct lensing             le;          /* for lensed spectra */
20     struct output              op;          /* for output files */
21     ErrorMessage errmsg;          /* for error messages */
22 };
23
24
25 struct Cosmology
26 {
27
28     struct Class_Cosmology_Struct ccs;
29     struct Line **Lines;
30
31     int NLines;
32     long mode_nu;
33 }
  
```

```

41
42     double cosmo_pars[6];
43 };
44
45
46
47 struct Line
48 {
49     long          LineType;
50     int           initialized;
51     size_t        npointsInterp;
52
53     double        line_freq;
54
55     gsl_interp_accel *mom1_accel_ptr;
56     gsl_spline       *mom1_spline_ptr;
57     gsl_interp_accel *mom2_accel_ptr;
58     gsl_spline       *mom2_spline_ptr;
59
60     gsl_interp_accel *b1_LW_accel_ptr;
61     gsl_spline       *b1_LW_spline_ptr;
62     gsl_interp_accel *b2_LW_accel_ptr;
63     gsl_spline       *b2_LW_spline_ptr;
64
65 };
66
67 struct globals
68 {
69     double H0;
70     double c;
71
72     double As;
73     double logAs;
74     double ns;
75     double h;
76     double Omega_cdm;
77     double Omega_b;
78     double Omega_r;
79     double Omega_lambda;
80     double Omega_g;
81     double Omega_nu;
82
83     double b1;
84     double sigFOG0;
85
86     long Npars;
87     double z_i;
88     double rho;
89     double mass;
90     double kp;
91     double ng;
92     double volume;
93
94     double kf;
95     double h_m;
96
97     double M_min;
98     double M_max;
99     double z_max;
100
101     char project_home[FILENAME_MAX];
102     char output_dir[FILENAME_MAX];
103     char data_dir[FILENAME_MAX];
104     char data_priors[FILENAME_MAX];
105
106     // Min and max values
107     double PS_kmin;
108     double PS_kmax;
109
110     // File names
111     char SFR_filename[FILENAME_MAX];
112     char Planck_Fisher_filename[FILENAME_MAX];
113
114     gsl_interp_accel *logM_accel_ptr;
115     gsl_interp_accel *z_accel_ptr;
116     gsl_spline2d     *logSFR_spline2d_ptr;
117 };
118
119 #endif
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134

```

```

135
136
137
138
139
140
141
142
143
144
145
146
147

```

5.4 header.h File Reference

```

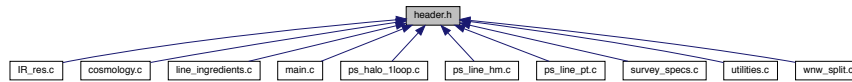
#include <time.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <float.h>
#include <string.h>
#include <omp.h>
#include <mpi.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_spline.h>
#include <gsl/gsl_interp2d.h>
#include <gsl/gsl_spline2d.h>
#include <gsl/gsl_sf_bessel.h>
#include <gsl/gsl_sf_legendre.h>
#include <gsl/gsl_integration.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_monte.h>
#include <gsl/gsl_monte_vegas.h>
#include <gsl/gsl_odeiv2.h>
#include <gsl/gsl_roots.h>
#include <gsl/gsl_sf_expint.h>
#include <ctype.h>
#include "../Class/include/class.h"
#include "cuba.h"
#include "Global_Structs.h"
#include "cosmology.h"
#include "utilities.h"
#include "survey_specs.h"
#include "primordial.h"
#include "line_ingredients.h"
#include "wnw_split.h"
#include "IR_res.h"
#include "ps_halo_lloop.h"
#include "ps_line_pt.h"
#include "ps_line_hm.h"
#include "cubature.h"

```

Include dependency graph for header.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [integrand_parameters](#)
A structure passed to the integrators to hold the parameters fixed in the integration.
- struct [integrand_parameters2](#)
Another structure passed to the integrators to hold the parameters fixed in the integration.

Macros

- `#define _GNU_SOURCE`
- `#define PSC 101L`
For solving ODER.
- `#define ST 102L`
- `#define TR 103L`
- `#define GROWTH 104L`
- `#define DERGROWTH 105L`
- `#define NONLINEAR 106L`
- `#define LINEAR 107L`
- `#define GAUSSIAN 114L`
- `#define NONGAUSSIAN 115L`
- `#define INIT 116L`
- `#define LOCAL 117L`
- `#define EQUILATERAL 118L`
- `#define ORTHOGONAL 119L`
- `#define QSF 120L`
- `#define HS 121L`
- `#define NGLOOP 122L`
- `#define derNGLOOP 123L`
- `#define QUADRATIC 124L`
- `#define TIDE 125L`
- `#define GAMMA 126L`
- `#define LPOWER 127L`
- `#define NLPOWER 128L`
- `#define TRANS 129L`
- `#define DER 130L`
- `#define CO10 131L`
- `#define CO21 132L`
- `#define CO32 133L`
- `#define CO43 134L`
- `#define CO54 135L`
- `#define CO65 136L`
- `#define CII 137L`
- `#define MATTER 138L`
- `#define LINEMATTER 139L`

- `#define` `LINE` 140L
- `#define` `DST` 141L
- `#define` `GFILTER` 142L
- `#define` `BSPLINE` 143L
- `#define` `TREE` 144L
- `#define` `LOOP` 145L
- `#define` `WIR` 146L
- `#define` `NOIR` 147L
- `#define` `HALO` 148L
- `#define` `PS_KMIN` 1.0e-7
- `#define` `PS_KMAX` 1.0e4
- `#define` `CLEANUP` 1
- `#define` `DO_NOT_EVALUATE` -1.0
- `#define` `MAXL` 2000

Functions

- void `initialize` ()
List of `limHaloPT` header files.
- void `cleanup` ()

5.4.1 Macro Definition Documentation

5.4.1.1 `_GNU_SOURCE` `#define` `_GNU_SOURCE`

5.4.1.2 `BSPLINE` `#define` `BSPLINE` 143L

5.4.1.3 `CII` `#define` `CII` 137L

5.4.1.4 `CLEANUP` `#define` `CLEANUP` 1

5.4.1.5 `CO10` `#define` `CO10` 131L

5.4.1.6 `CO21` `#define` `CO21` 132L

5.4.1.7 CO32 `#define CO32 133L`

5.4.1.8 CO43 `#define CO43 134L`

5.4.1.9 CO54 `#define CO54 135L`

5.4.1.10 CO65 `#define CO65 136L`

5.4.1.11 DER `#define DER 130L`

5.4.1.12 DERGROWTH `#define DERGROWTH 105L`

5.4.1.13 derNGLOOP `#define derNGLOOP 123L`

5.4.1.14 DO_NOT_EVALUATE `#define DO_NOT_EVALUATE -1.0`

5.4.1.15 DST `#define DST 141L`

5.4.1.16 EQUILATERAL `#define EQUILATERAL 118L`

5.4.1.17 GAMMA `#define GAMMA 126L`

5.4.1.18 GAUSSIAN #define GAUSSIAN 114L

5.4.1.19 GFILTER #define GFILTER 142L

5.4.1.20 GROWTH #define GROWTH 104L

5.4.1.21 HALO #define HALO 148L

5.4.1.22 HS #define HS 121L

5.4.1.23 INIT #define INIT 116L

5.4.1.24 LINE #define LINE 140L

5.4.1.25 LINEAR #define LINEAR 107L

5.4.1.26 LINEMATTER #define LINEMATTER 139L

5.4.1.27 LOCAL #define LOCAL 117L

5.4.1.28 LOOP #define LOOP 145L

5.4.1.29 LPOWER `#define LPOWER 127L`

5.4.1.30 MATTER `#define MATTER 138L`

5.4.1.31 MAXL `#define MAXL 2000`

5.4.1.32 NGLOOP `#define NGLOOP 122L`

5.4.1.33 NLPOWER `#define NLPOWER 128L`

5.4.1.34 NOIR `#define NOIR 147L`

5.4.1.35 NONGAUSSIAN `#define NONGAUSSIAN 115L`

5.4.1.36 NONLINEAR `#define NONLINEAR 106L`

5.4.1.37 ORTHOGONAL `#define ORTHOGONAL 119L`

5.4.1.38 PS_KMAX `#define PS_KMAX 1.0e4`

5.4.1.39 PS_KMIN `#define PS_KMIN 1.0e-7`

5.4.1.40 PSC `#define PSC 101L`

For solving ODER.

5.4.1.41 QSF `#define QSF 120L`

5.4.1.42 QUADRATIC `#define QUADRATIC 124L`

5.4.1.43 ST `#define ST 102L`

5.4.1.44 TIDE `#define TIDE 125L`

5.4.1.45 TR `#define TR 103L`

5.4.1.46 TRANS `#define TRANS 129L`

5.4.1.47 TREE `#define TREE 144L`

5.4.1.48 WIR `#define WIR 146L`

5.4.2 Function Documentation

5.4.2.1 cleanup() `void cleanup ()`

5.4.2.2 initialize() `void initialize ()`

List of limHaloPT header files.

Function declarations of [main.c](#) module

List of limHaloPT header files.

The global structure "gb" have several elements to hold the paths to project source directory, input, and output folders, and values of cosmological parameters.

Returns

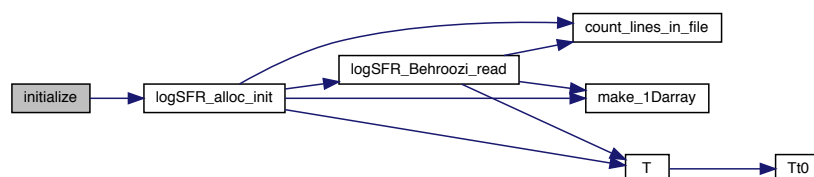
void

Change the path to the parent directory

In units of km/s

$\omega_b = \Omega_b h^2$;

3.0665 Here is the call graph for this function:



Here is the caller graph for this function:



5.5 header.h

[Go to the documentation of this file.](#)

```

1
2
3
4
5
6 #ifndef HEADER_H_
7 #define HEADER_H_
8
9 #define _GNU_SOURCE
10
11 #include <time.h>
12 #include <unistd.h>
13 #include <stdlib.h>
14 #include <stdio.h>
15 #include <math.h>
16 #include <float.h>
17 #include <string.h>
18 #include <omp.h>
19 #include <mpi.h>
20 #include <gsl/gsl_errno.h>
21 #include <gsl/gsl_spline.h>
22 #include <gsl/gsl_interp2d.h>
23 #include <gsl/gsl_spline2d.h>
24 #include <gsl/gsl_sf_bessel.h>
25 #include <gsl/gsl_sf_legendre.h>
26 #include <gsl/gsl_integration.h>
27 #include <gsl/gsl_matrix.h>
28 #include <gsl/gsl_linalg.h>
29 #include <gsl/gsl_blas.h>
30 #include <gsl/gsl_monte.h>
31 #include <gsl/gsl_monte_vegas.h>
32 #include <gsl/gsl_odeiv2.h>
33 #include <gsl/gsl_roots.h> // For finding the root of algebraic equation
34 #include <gsl/gsl_sf_expint.h>
35 #include <ctype.h>
36 #include "../Class/include/class.h"
37 #include "cuba.h"
38
39
40 #define PSC          101L
41 #define ST           102L
42 #define TR           103L
43
44 #define GROWTH       104L
45 #define DERGROWTH   105L
46
47 #define NONLINEAR    106L
48 #define LINEAR       107L
49
50 #define GAUSSIAN      114L
51 #define NONGAUSSIAN  115L
52
53 #define INIT         116L
54 #define LOCAL        117L
55 #define EQUILATERAL  118L
56 #define ORTHOGONAL   119L
57 #define QSF          120L
58 #define HS           121L
59 #define NGLOOP       122L
60 #define derNGLOOP    123L
61
62 #define QUADRATIC     124L
63 #define TIDE          125L
64 #define GAMMA         126L
65
66 #define LPOWER        127L
67 #define NLPOWER       128L
68 #define TRANS         129L
69 #define DER           130L
70
71 #define CO10          131L
72 #define CO21          132L
73 #define CO32          133L
74 #define CO43          134L
75 #define CO54          135L
76 #define CO65          136L
77 #define CII           137L
78
79 #define MATTER        138L
80 #define LINEMATTER    139L
81 #define LINE          140L
82
83 #define DST           141L
84 #define GFILTER       142L
85 #define BSPLINE       143L
86

```

```
87
88 #define TREE          144L
89 #define LOOP          145L
90 #define WIR           146L
91 #define NOIR          147L
92
93 #define HALO           148L
94
95
96 #define PS_KMIN        1.0e-7
97 #define PS_KMAX        1.0e4
98
99 #define CLEANUP        1
100
101 #define DO_NOT_EVALUATE -1.0
102
103 #define MAXL 2000
104
105 #include "Global_Structs.h"
106 #include "cosmology.h"
107 #include "utilities.h"
108 #include "survey_specs.h"
109 #include "primordial.h"
110 #include "line_ingredients.h"
111 #include "wnw_split.h"
112 #include "IR_res.h"
113 #include "ps_halo_lloop.h"
114 #include "ps_line_pt.h"
115 #include "ps_line_hm.h"
116 #include "cubature.h"
117
118
119 void initialize();
120 void cleanup();
121
122
123 struct integrand_parameters
124 {
125     double p1;
126     double p2;
127     double p3;
128     double p4;
129     double p5;
130     double p6;
131     double p7;
132     double p8;
133     double p9;
134     double p10;
135     double p11;
136     long p12;
137     long p13;
138 };
139
140 struct integrand_parameters2
141 {
142     struct Cosmology *p1;
143     struct Cosmology *p2;
144     struct Cosmology *p3;
145
146     double p4;
147     double p5;
148     double p6;
149     double p7;
150     double p8;
151     double p9;
152     double p10;
153     double p11;
154     double p12;
155
156     long p13;
157     long p14;
158     long p15;
159     long p16;
160     long p17;
161     long p18;
162
163     int p19;
164
165     double *p20;
166     size_t p22;
167 };
168
169 #endif
170
171
```

186
187
188
189
190

5.6 cosmology.c File Reference

Documented cosmology module.

```
#include "header.h"
```

Include dependency graph for cosmology.c:



Functions

- int [Cosmology_init](#) (struct [Cosmology](#) *Cx, double pk_kmax, double pk_zmax, int nlines, int *line_types, size_t npoints_interp, double M_min, long mode_mf)
Allocate memory and initialize the cosmology structure, which includes the CLASS cosmology structure and line structure.
- int [Cosmology_free](#) (struct [Cosmology](#) *Cx)
Free the memory allocated to cosmology structure.
- int [CL_Cosmology_initialize](#) (struct [Cosmology](#) *Cx, double pk_kmax, double pk_zmax)
Allocate memory and initialize the CLASS cosmology structure.
- int [CL_Cosmology_free](#) (struct [Cosmology](#) *Cx)
Free the memory allocated to CLASS cosmology structure.
- double [Pk_dlnPk](#) (struct [Cosmology](#) *Cx, double k, double z, int mode)
Compute the matter power spectra (in unit of $(\text{Mpc})^3$) as a function of k (in unit of $1/\text{Mpc}$) and z , Setting the switch "mode", to LINEAR or NONLINEAR, we can compute the linear or nonlinear spectrum respectively.
- double [Pk_dlnPk_HV](#) (struct [Cosmology](#) *Cx, double k, double z, int mode)
Read in the linear power spectrum, used to set the initial conditions of Hidden-Valley sims.
- double [Mk_dlnMk](#) (struct [Cosmology](#) *Cx, double k, double z, int mode)
Compute the transfer function for different species depending on the switch "mode", which can be set to cdm, baryons or total matter transfer function.
- double [sig_sq_integrand](#) (double x, void *par)
The integrand function passed to qags integrator to compute the variance of the matter density.
- double [sig_sq](#) (struct [Cosmology](#) *Cx, double z, double R)
Compute variance of smoothed matter density fluctuations.
- double [der_Insig_sq](#) (struct [Cosmology](#) *Cx, double z, double R)
Compute the logarithmic derivative of the variance of smoothed matter density fluctuations w.r.t.
- double [sigma0_sq_integrand](#) (double x, void *par)
The integrand function passed to qags integrator to compute the variance of the unsmoothed matter density.
- double [sigma0_sq](#) (struct [Cosmology](#) *Cx, double z, double kmax)
Compute variance of unsmoothed matter density fluctuations.
- double [growth_D](#) (struct [Cosmology](#) *Cx, double z)
Compute the growth factor $D(k,z)$ which is scale-indep if mode_nu = NUM, and scale-dep if mode_nu = MASS The scale-dep growth is calculated by taking the ratio of the transfer function at redshift z and zero.
- double [growth_f](#) (struct [Cosmology](#) *Cx, double z)
Compute the scale-dependant linear growth rate $f(k,z)$ (i.e the velocity growth factor) by taking numerical derivative of the scale_dep_growth_D() function $f(k,a) = d \ln D(k,a) / d \ln a$.

- double [Hubble](#) (struct [Cosmology](#) *Cx, double z)
Compute the the hubble rate (exactly the quantity defined by CLASS as index_bg_H in the background module).
- double [angular_distance](#) (struct [Cosmology](#) *Cx, double z)
Compute the angular diameter distance (exactly the quantity defined by CLASS as ba.index_bg_ang_distance in the background module).
- double [comoving_radial_distance](#) (struct [Cosmology](#) *Cx, double z)
Compute the comoving radial distance
- double [rhoc](#) (struct [Cosmology](#) *Cx, double z)
Compute the critical density in unit of $M_{\text{sun}}/\text{Mpc}^3$.
- double [R_scale](#) (struct [Cosmology](#) *Cx, double M)
Compute the Lagrangian radius of halos in unit of $1/\text{Mpc}^3$, fixing $z=0$.
- double [R_vir](#) (struct [Cosmology](#) *Cx, double M)
Compute the comoving virial radius of halos in unit of $1/\text{Mpc}^3$, which is defined as the radius at which the average density within this radius is $\Delta \times \rho_c$.
- double [concentration_cdm](#) (double M, double z)
Compute the cold dark matter concentration-mass relation.
- double [nfw_profile](#) (struct [Cosmology](#) *Cx, double k, double M, double z)
Compute the NFW halo profile in Fourier space, given by Eq.
- double [window_rth](#) (double k, double R)
Fourier transform of top-hat window in real space.
- double [derR_window_rth](#) (double k, double R)
Derivative w.r.t.
- double [window_kth](#) (double k, double R)
Top-hat window in Fourier space.
- double [window_g](#) (double k, double R)
Gaussian window.
- double [derR_logwindow_g](#) (double k, double R)
Derivative w.r.t smoothing scale of Gaussian window.

Variables

- struct [globals](#) [gb](#)

5.6.1 Detailed Description

Documented cosmology module.

Azadeh Moradinezhad Dizgah, November 4th 2021

The first routine of this module initializes the [Cosmology](#) structure, which is the main building block of this entire code. This structure includes two sub-structures: the CLASS cosmology structure and line structure. Once the CLASS cosmology is initialized, various useful functions can be directly called from CLASS, example to compute matter power spectrum and transfer function, angular and comoving radii, growth factor and growth rate, variance of matter fluctuations and its derivative. Lastly, the module also includes various window functions and their derivatives.

In summary, the following functions can be called from other modules:

1. [Cosmology_init\(\)](#) allocates memory to and initializes cosmology structure
2. [Cosmology_free\(\)](#) frees the memory allocated to cosmology structure

3. `CL_Cosmology_initialize()` initializes the class cosmology structure
4. `CL_Cosmology_free()` frees the class cosmology structure
5. `PS()` computes matter power spectrum calling class function
6. `Transfer()` computes matter transfer function calling class function
7. `growth_D()` computes the scale-dep growth factor
8. `growth_f()` computes the scale-dep growth rate $d\ln D(k,a)/d\ln a$
9. `scale_indep_growth_D()` computes the scale-indep growth factor using directly CLASS functions
10. `scale_indep_growth_f()` computes the scale-indep growth rate $d\ln D(k,a)/d\ln a$ using directly CLASS functions
11. `Hubble()` computes hubble parameter using directly CLASS functions
12. `angular_distance()` computes angular diameter distance using directly CLASS functions
13. `comoving_radial_distance()` computes radial distance using directly CLASS functions
14. `sig_sq()` computes variance of smoothed matter fluctuations
15. `der_sig_sq()` computes derivative of the variance of smoothed matter fluctuations w.r.t. smoothing scale
16. `sigma0_sq()` computes variance of unsmoothed matter fluctuations
17. `rhoc()` computes the critical density of the universe
18. `R_scale()` computes the size of a spherical halo corresponding to a given mass at $z=0$
19. `R_scale_wrong()` computes the size of a spherical halo corresponding to a given mass at a given redshift
20. `window_rth()` computes top-hat filter in real space
21. `window_g()` computes Gaussian window
22. `window_kth()` computes top-hat filter in Fourier space
23. `derR_window_rth()` computes derivative of top-hat filter in real space w.r.t. smoothing scale
24. `derR_logwindow_g()` computes derivative of top-hat filter in Fourier space w.r.t. smoothing scale

5.6.2 Function Documentation

5.6.2.1 angular_distance() `double angular_distance (`
`struct Cosmology * Cx,`
`double z)`

Compute the angular diameter distance (exactly the quantity defined by CLASS as `ba.index_bg_ang_distance` in the background module).

luminosity distance $d_L = (1+z) d_M$ angular diameter distance $d_A = d_M/(1+z)$ where d_M is the transverse comoving distance, which is equal to comoving distance for flat cosmology and has a dependance on curvature for non-flat cosmologies, as described in lines 849 - 851

Parameters

<i>Cx</i>	Input↔ : pointer to Cosmology struc- ture
<i>z</i>	Input↔ : red- shift to com- pute the spec- trum

Returns

D_A

junkHere is the caller graph for this function:



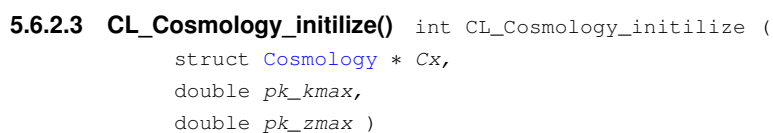
5.6.2.2 CL_Cosmology_free() `int CL_Cosmology_free (struct Cosmology * Cx)`

Free the memory allocated to CLASS cosmology structure.

Parameters

<i>Cx</i>	Input↔ : pointer to Cosmology struc- ture
-----------	---

Here is the caller graph for this function:



Parameters

Generated by Doxygen

Returns

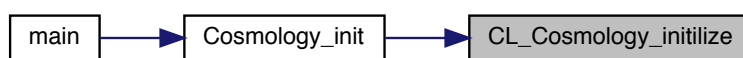
the error status

h

Omega_b

Omega_b

pivot scale in unit of 1/MpcHere is the caller graph for this function:



5.6.2.4 comoving_radial_distance() `double comoving_radial_distance (`
 `struct Cosmology * Cx,`
 `double z)`

Compute the comoving radial distance

Parameters

<i>Cx</i>	Input↔ : pointer to Cosmology struc- ture
<i>z</i>	Input↔ : red- shift to com- pute the spec- trum

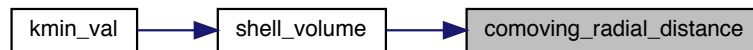
Returns

the double value D_c

junk

For a flat cosmology, comoving distance is equal to conformal distance. This piece of code is how the comoving distance for flat and nonflat cases are computed. Change the expression of `D_A` below according to this if considering non-flat cosmology.

Here is the caller graph for this function:



5.6.2.5 concentration_cdm() `double concentration_cdm (`
`double M ,`
`double z)`

Compute the cold dark matter concentration-mass relation.

Parameters

M	Input↔ : halo mass in unit of solar mass
z	Input↔ : red- shift of inter- est

Returns

the cdm concentration

Here is the caller graph for this function:




```

5.6.2.7 Cosmology_init()  int Cosmology_init (
    struct Cosmology * Cx,
    double pk_kmax,
    double pk_zmax,
    int nlines,
    int * line_types,
    size_t npoints_interp,
    double M_min,
    long mode_mf )

```

Allocate memory and initialize the cosmology structure, which includes the CLASS cosmology structure and line structure.

Parameters

<i>Cx</i>	Input↔ : pointer to Cosmology struc- ture
<i>pk_kmax</i>	Input↔ : kmax for com- puta- tion of matter power spec- trum by CLASS
<i>pk_zmax</i>	Input↔ : zmax for com- puta- tion of matter power spec- trum by CLASS
<i>nlines</i>	Input↔ : num- ber of lines whose prop- erties we want to com- pute

Parameters

<i>line_type</i>	Input↔ : name of the line to com- pute. It can be set to CII, CO10, CO21, CO32, CO43, CO54, CO65
<i>npoints_interp</i>	Input↔ : num- ber of points in red- shift for in- terpo- lation of line prop- erties
<i>M_min</i>	Input↔ : min- imum halo mass for mass inte- grals

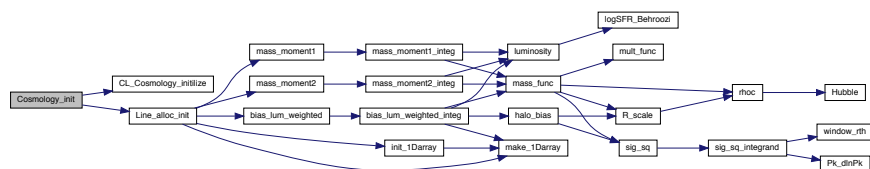
Parameters

<i>mode_mf</i>	Input ← : theoretical model of halo mass function to use. It can be set to sheth- ← Tormen (ST), Tinker (TR) or Press- ← Schechter (PSC)
----------------	---

Returns

an integer if succeeded

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.8 der_Insig_sq() `double der_Insig_sq (`
`struct Cosmology * Cx,`
`double z,`
`double R)`

Compute the logarithmic derivative of the variance of smoothed matter density fluctuations w.r.t.

smoothing scale

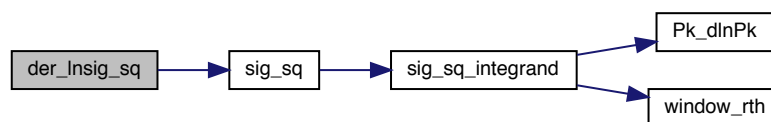
Parameters

<i>Cx</i>	Input↔ : pointer to Cosmology struc- ture
<i>z</i>	Input↔ : red- shift to com- pute the spec- trum
<i>R</i>	Input↔ : smooth- ing scale in unit of Mpc

Returns

the log-derivative of variance

Here is the call graph for this function:



5.6.2.9 derR_logwindow_g() `double derR_logwindow_g (`
`double k,`
`double R)`

Derivative w.r.t smoothing scale of Gaussian window.

Parameters

k	Input↔ : wavenum- ber in unit of 1/Mpc
R	Input↔ : smooth- ing scale in unit of Mpc

Returns

the derivative of the window function

5.6.2.10 `derR_window_rth()` `double derR_window_rth (`
 `double k,`
 `double R)`

Derivative w.r.t.

smoothing scale of the Fourier transform of top-hat window in real space

Parameters

k	Input↔ : wavenum- ber in unit of 1/Mpc
R	Input↔ : smooth- ing scale in unit of Mpc

Returns

the derivative of the window function

5.6.2.11 growth_D() `double growth_D (`
`struct Cosmology * Cx,`
`double z)`

Compute the growth factor $D(k,z)$ which is scale-indep if `mode_nu = NUM`, and scale-dep if `mode_nu = MASS`. The scale-dep growth is calculated by taking the ratio of the transfer function at redshift z and zero.

The scale-indep growth is computed by CLASS directly. The switch "mode" can be set to CDM, BA, TOT to return the growth factor of cdm, baryon and total matter.

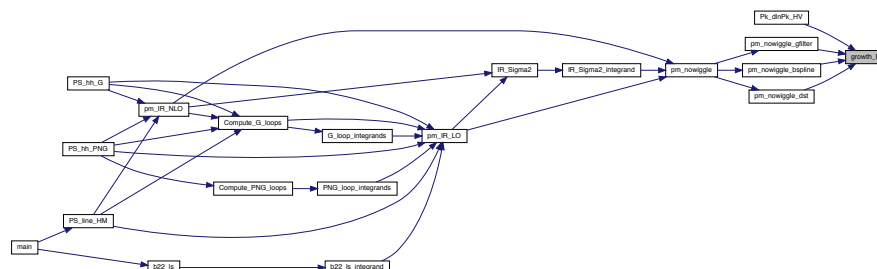
Parameters

<code>Cx</code>	Input↔ : pointer to Cosmology struc- ture
<code>k</code>	Input↔ : wavenumb- er in unit of 1/Mpc
<code>z</code>	Input↔ : red- shift to com- pute the spec- trum

Returns

the growth factor, can be k-dep (ex. with nonzero neutrino mass)

junkHere is the caller graph for this function:



5.6.2.12 growth_f() `double growth_f (`
`struct Cosmology * Cx,`
`double z)`

Compute the scale-dependant linear growth rate $f(k,z)$ (i.e the velocity growth factor) by taking numerical derivative of the `scale_dep_growth_D()` function $f(k,a) = d \ln D(k,a) / d \ln a$.

The switch "mode" can be set to CDM, BA, TOT to return the growth factor of the corresponding matter component.

This is a useful function when constraining physics that induces scale-dependant growth such as massive neutrinos.

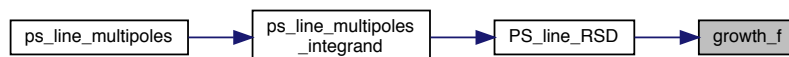
Parameters

<code>Cx</code>	Input↔ : pointer to Cosmology struc- ture
<code>k</code>	Input↔ : wavenumb- ber in unit of 1/Mpc
<code>z</code>	Input↔ : red- shift to com- pute the spec- trum

Returns

the growth rate, can be k-dep (ex. with nonzero neutrino mass)

junkHere is the caller graph for this function:



5.6.2.13 Hubble() `double Hubble (`
`struct Cosmology * Cx,`
`double z)`

Compute the the hubble rate (exactly the quantity defined by CLASS as `index_bg_H` in the background module).

This function is to a good approximation equal to $Hubble(a,Cx) = gb.h * \sqrt{Eofa(a,Cx)}$

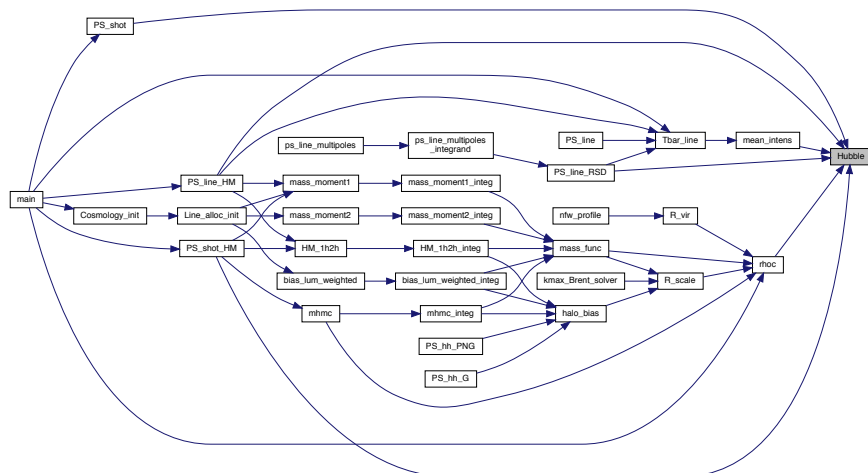
Parameters

Cx	Input↔ : pointer to Cosmology struc- ture
z	Input↔ : red- shift to com- pute the spec- trum

Returns

the hubble parameter

junkHere is the caller graph for this function:



```
5.6.2.14 Mk_dlnMk() double Mk_dlnMk (
    struct Cosmology * Cx,
    double k,
    double z,
    int mode )
```

Compute the transfer function for different species depending on the switch "mode", which can be set to cdm, baryons or total matter transfer function.

CLASS function spectra_tk_at_k_and_z() routine evaluates the matter transfer functions at a given value of k and z by interpolating in a table of all $T_i(k, z)$'s computed at this z by spectra_tk_at_z() (when $k_{min} \leq k \leq k_{max}$). Returns an error when $k < k_{min}$ or $k > k_{max}$.

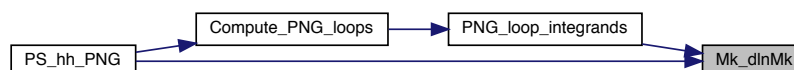
Parameters

Cx	Input↔ : pointer to Cosmology struc- ture
k	Input↔ : wavenumb- ber in unit of 1/Mpc
z	Input↔ : red- shift to com- pute the spec- trum
$mode$	Input↔ : switch to de- cide for which species we want to get the trans- fer func- tion

Returns

the transfer function

Here is the caller graph for this function:



5.6.2.15 nfw_profile() `double nfw_profile (`
`struct Cosmology * Cx,`
`double k,`
`double M,`
`double z)`

Compute the NFW halo profile in Fourier space, given by Eq.

3.7 of 2004.09515 The profile is normalized to unity at $k \rightarrow 0$, (see fig 3 of 1003.4740)

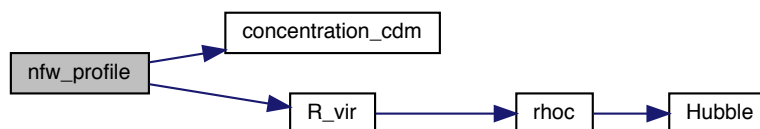
Parameters

<i>Cx</i>	Input↔ : pointer to Cosmology struc- ture
<i>k</i>	Input↔ : wavenum- ber in unit of 1/Mpc
<i>M</i>	Input↔ : halo mass in unit of solar mass
<i>z</i>	Input↔ : red- shift of inter- est

Returns

the nfw profile

ρ_s is computed by enforcing $\int dr r^2 u(r) = 1$ Here is the call graph for this function:



```
5.6.2.16 Pk_dlnPk() double Pk_dlnPk (
    struct Cosmology * Cx,
    double k,
    double z,
    int mode )
```

Compute the matter power spectra (in unit of $(\text{Mpc})^3$) as a function of k (in unit of $1/\text{Mpc}$) and z , Setting the switch "mode", to LINEAR or NONLINEAR, we can compute the linear or nonlinear spectrum respectively.

The CLASS `spectra_pk_at_k_and_z()` and `spectra_pk_nl_at_k_and_z`, evaluate the matter power spectrum at a given value of k and z by interpolating in a table of all $P(k)$'s computed at this z by `spectra_pk_at_z()` (when $k_{\min} \leq k \leq k_{\max}$), or eventually by using directly the primordial spectrum (when $0 \leq k < k_{\min}$): the latter case is an approximation, valid when $k_{\min} \ll$ comoving Hubble scale today. Returns zero when $k=0$. Returns an error when $k < 0$ or $k > k_{\max}$.

Parameters

Cx	Input↔ : pointer to Cosmology struc- ture
k	Input↔ : wavenumb- ber in unit of 1/Mpc
z	Input↔ : red- shift to com- pute the spec- trum


```
5.6.2.17 Pk_dlnPk_HV() double Pk_dlnPk_HV (
    struct Cosmology * Cx,
    double k,
    double z,
    int mode )
```

Read in the linear power spectrum, used to set the initial conditions of Hidden-Valley sims.

Input k is in unit of 1/Mpc. First convert it to h/Mpc, and also convert the final matter power spectrum in unit of $(\text{Mpc}/h)^3$

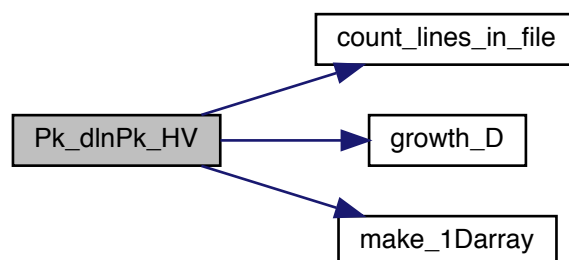
Parameters

<i>Cx</i>	Input↔ : pointer to <i>Cosmology</i> struc- ture
<i>k</i>	Input↔ : wavenumb- ber in unit of 1/Mpc
<i>z</i>	Input↔ : red- shift to com- pute the spec- trum
<i>mode</i>	Input↔ : switch to de- cide whether to eval- uate the inter- polator of the power spec- trum or free the inter- polator

Returns

the HV linear matter power spectrum

Here is the call graph for this function:



5.6.2.18 R_scale() `double R_scale (`
 `struct Cosmology * Cx,`
 `double M)`

Compute the Lagrangian radius of halos in unit of $1/\text{Mpc}^3$, fixing $z=0$.

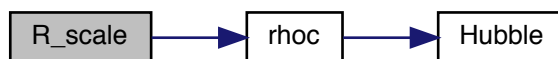
Parameters

<code>Cx</code>	Input↔ : pointer to <code>Cosmology</code> struc- ture
<code>h_mass</code>	Input↔ : halo mass in unit of solar mass

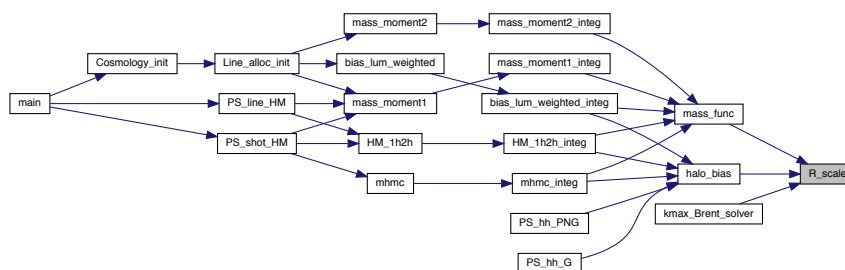
Returns

R_s

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.6.2.19 R_vir() double R_vir (
    struct Cosmology * Cx,
    double M )
  
```

Compute the comoving virial radius of halos in unit of $1/\text{Mpc}^3$, which is defined as the radius at which the average density within this radius is $\Delta \times \rho_c$.

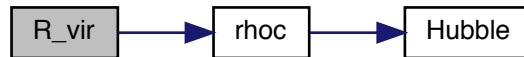
Parameters

<i>Cx</i>	Input↔ : pointer to <code>Cosmology</code> structure
<i>M</i>	Input↔ : halo mass in unit of solar mass

Returns

R_vir

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.20 rhoc() `double rhoc (`
 `struct Cosmology * Cx,`
 `double z)`

Compute the critical density in unit of $M_{\text{sun}}/\text{Mpc}^3$.

Parameters

<code>Cx</code>	Input↔ : pointer to <code>Cosmology</code> struc- ture
<code>z</code>	Input↔ : red- shift to com- pute the spec- trum

Returns

the double value of rho_c

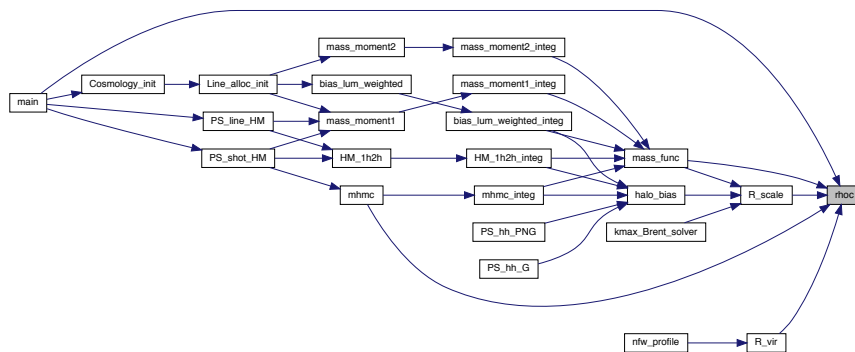
$$E(a) = H(a)^2 / H_0^2$$

G is in unit of $m^3 \text{ kg}^{-1} \text{ s}^{-2}$, conversion factor from m to Mpc

To convert to solar mass Here is the call graph for this function:



Here is the caller graph for this function:



```

5.6.2.21 sig_sq() double sig_sq (
    struct Cosmology * Cx,
    double z,
    double R )
  
```

Compute variance of smoothed matter density fluctuations.

The function `sig_sq_integrand()` defines the integrand and `sig_sq()` computes the k-integral

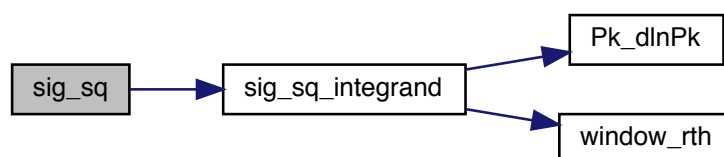
Parameters

Cx	Input↔ : pointer to Cosmology struc- ture
z	Input↔ : red- shift to com- pute the spec- trum
R	Input↔ : smoothing scale in unit of Mpc

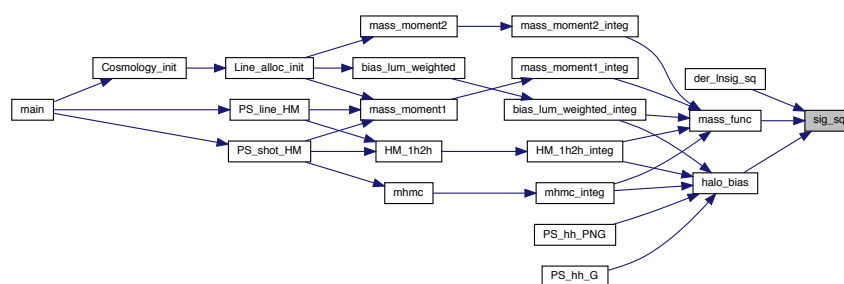
Returns

the variance

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.22 sig_sq_integrand() `double sig_sq_integrand (`
`double x,`
`void * par)`

The integrand function passed to qags integrator to compute the variance of the matter density.

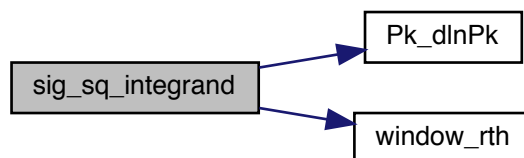
Parameters

<i>x</i>	Input↵ : inte- gration vari- able
<i>par</i>	Input↵ : inte- gration par- maeters

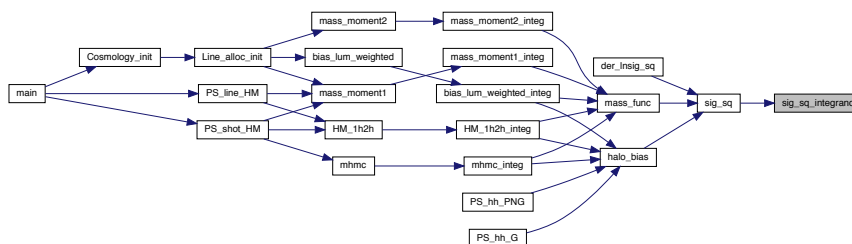
Returns

value of the integrand

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.23 sigma0_sq() double sigma0_sq (
 struct [Cosmology](#) * Cx,
 double z,
 double kmax)

Compute variance of unsmoothed matter density fluctuations.

The function sigma0_integrand() defines the integrand and [sigma0_sq\(\)](#) computes the k-integral

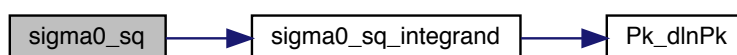
Parameters

Cx	Input↔ : pointer to Cosmology struc- ture
z	Input↔ : red- shift to com- pute the spec- trum

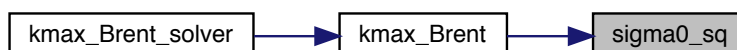
Returns

the unsmoothed variance kmax is in unit of 1/Mpc

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.6.2.24 sigma0_sq_integrand() double sigma0_sq_integrand (  
    double x,  
    void * par )
```

The integrand function passed to qags integrator to compute the variance of the unsmoothed matter density.

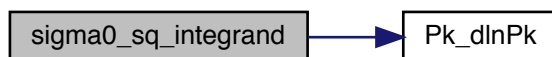
Parameters

<i>x</i>	Input↔ : inte- gration vari- able
<i>par</i>	Input↔ : inte- gration par- maeters

Returns

value of the integrand

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.6.2.25 window_g() double window_g (  
    double k,  
    double R )
```

Gaussian window.

Parameters

k	Input↔ : wavenum- ber in unit of 1/Mpc
R	Input↔ : smooth- ing scale in unit of Mpc

Returns

the window function

5.6.2.26 window_kth() `double window_kth (`
`double k ,`
`double R)`

Top-hat window in Fourier space.

Parameters

k	Input↔ : wavenum- ber in unit of 1/Mpc
R	Input↔ : smooth- ing scale in unit of Mpc

Returns

the window function

5.6.2.27 window_rth() `double window_rth (`
`double k ,`
`double R)`

Fourier transform of top-hat window in real space.

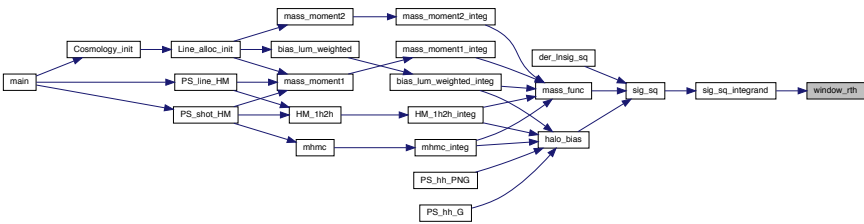
Parameters

<i>k</i>	Input↔ : wavenum- ber in unit of 1/Mpc
<i>R</i>	Input↔ : smoothing scale in unit of Mpc

Returns

the window function

Here is the caller graph for this function:



5.6.3 Variable Documentation

5.6.3.1 gb struct globals gb

5.7 IR_res.c File Reference

Documented IR_res module.

#include "header.h"
Include dependency graph for IR_res.c:



Functions

- double [pm_IR_LO](#) (struct [Cosmology](#) *Cx, double k, double z, long SPLIT)
Compute the leading-order IR-resummed matter power spectrum, ala Ivanovic et al.
- double [pm_IR_NLO](#) (struct [Cosmology](#) *Cx, double k, double z, long SPLIT)
Compute the next-to-leading-order IR-resummed matter power spectrum, ala Ivanovic et al.
- double [IR_Sigma2_integrand](#) (double x, void *par)
Integrand to compute the suppression factor IR_sigma2.
- double [IR_Sigma2](#) (struct [Cosmology](#) *Cx, double z, double kf0, long SPLIT)
Compute the suppression factor IR_sigma2.
- double [pm_nowiggle](#) (struct [Cosmology](#) *Cx, double k, double z, double kf0, int [cleanup](#), long SPLIT)
Compute the no-wiggle componenet of the matter power spectrum.
- double [pm_nowiggle_bspline](#) (struct [Cosmology](#) *Cx, double k, double z, int [cleanup](#))
Compute the no-wiggle componenet of the matter power spectrum, reading in and interpolating the output of apython code which computed the broadband by fitting families of Bsplines (see Vlah et al 2015)
- double [pm_nowiggle_gfilter](#) (struct [Cosmology](#) *Cx, double k, double z, int [cleanup](#))
Compute the no-wiggle componenet of the matter power spectrum, using Gaussian filter (see Vlah et al 2015)
- double [pm_nowiggle_dst](#) (struct [Cosmology](#) *Cx, double k, double z, int [cleanup](#))
Compute the no-wiggle componenet of the matter power spectrum, reading in and interpolating the output of apython code which computed the broadband by discrete sin-transform, See Hamann et al 2010.

5.7.1 Detailed Description

Documented IR_res module.

Azadeh Moradinezhad Dizgah, November 4th 2021

This module is computes the leading and next-to-leading IR-resummed matter power spectrum The wiggle-nowiggle seperation is performed in [wnw_split.c](#) module.

In summary, the following functions can be called from other modules:

1. [pm_IR_LO\(\)](#)
2. [pm_IR_NLO\(\)](#)
3. [IR_Sigma2\(\)](#)
4. [pm_nowiggle\(\)](#)
5. [pm_nowiggle_gfilter\(\)](#)
6. [pm_nowiggle_bspline\(\)](#)
7. [pm_nowiggle_dst\(\)](#)

5.7.2 Function Documentation

5.7.2.1 IR_Sigma2() double IR_Sigma2 (
 struct [Cosmology](#) * Cx,
 double z,
 double kf0,
 long SPLIT)

Compute the suppression factor IR_sigma2.

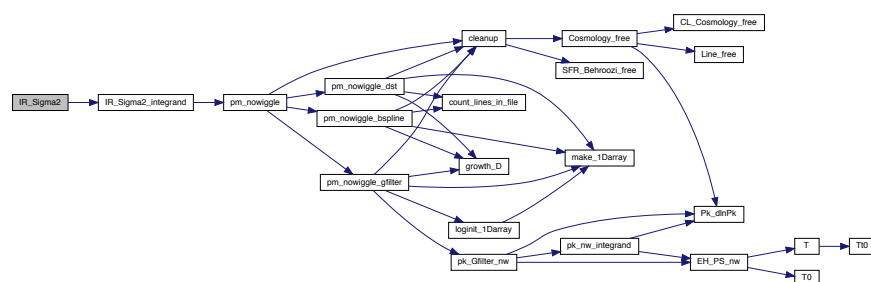
Parameters

<i>Cx</i>	Input↔ : pointer to cos- mol- ogy struc- ture
<i>z</i>	Input↔ : red- shift
<i>kf0</i>	Input↔ : first ele- ment of the k- array, used in nor- mal- ization of EH no- wiggle spec- trum
<i>SPLIT</i>	Input↔ : switch to set the method of wiggle- nowiggle split

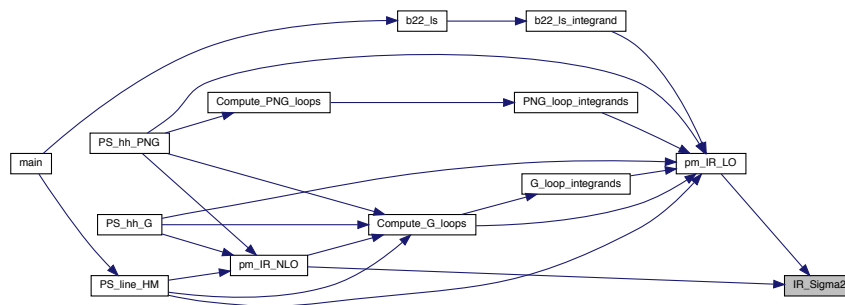
Returns

value of IR resummation suppression factor

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.2.2 IR_Sigma2_integrand() `double IR_Sigma2_integrand (`
`double x,`
`void * par)`

Integrand to compute the suppression factor IR_sigma2.

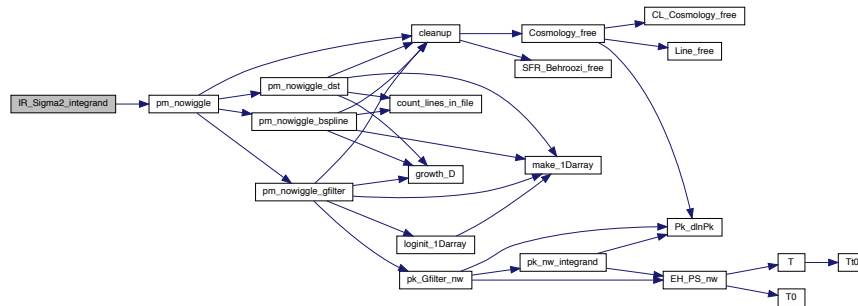
Parameters

<i>x</i>	Input↵ : inte- gration vari- able, k- values
<i>par</i>	Input↵ : inte- gration param- eters

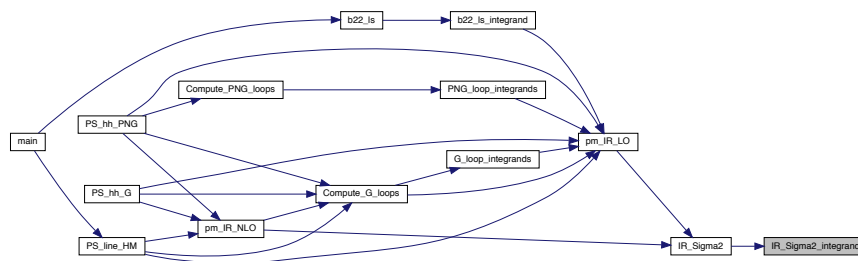
Returns

integrand to be used in IR_sigma2() function

BAO_scale = 110. Mpc/h. Here is the call graph for this function:



Here is the caller graph for this function:



```

5.7.2.3 pm_IR_LO() double pm_IR_LO (
    struct Cosmology * Cx,
    double k,
    double z,
    long SPLIT )
  
```

Compute the leading-order IR-resummed matter power spectrum, ala Ivanovic et al.

Parameters

Cx	Input↔ : pointer to cos- mol- ogy struc- ture
----	--

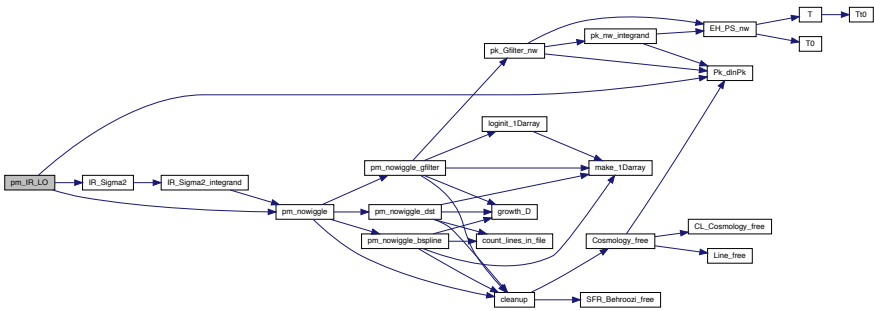
Parameters

<i>k</i>	Input↔ : wavenum- ber in unit of 1/Mpc.
<i>z</i>	Input↔ : red- shift
<i>SPLIT</i>	Input↔ : switch to set the method of wiggle- nowiggle split

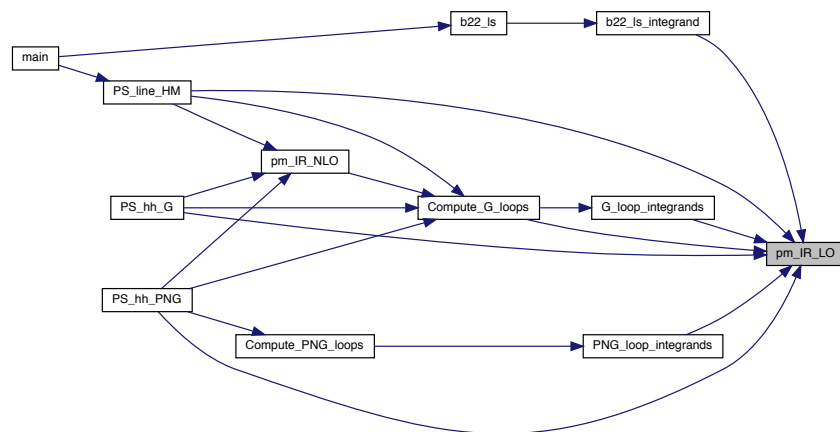
Returns

value of leading IR-ressumed power spectrum

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.2.4 pm_IR_NLO() double pm_IR_NLO (

```

    struct Cosmology * Cx,
    double k,
    double z,
    long SPLIT )

```

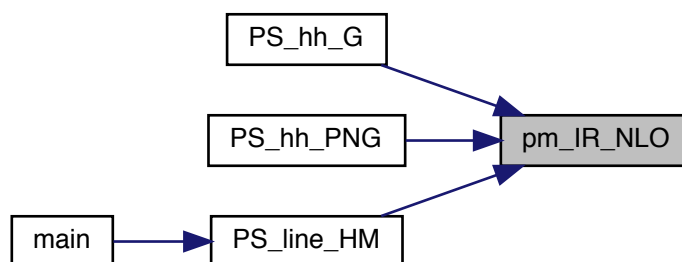
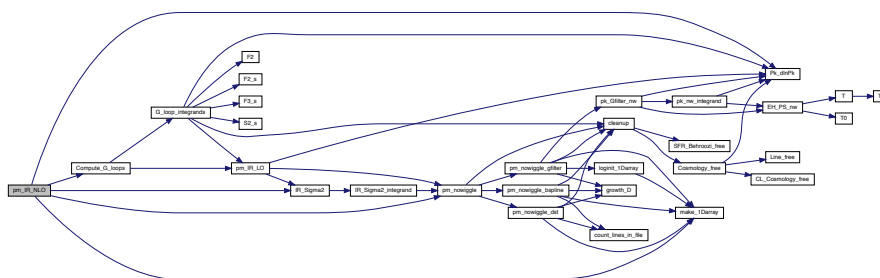
Compute the next-to-leading-order IR-resummed matter power spectrum, ala Ivanovic et al.

Parameters

Cx	Input↔ : pointer to cos- mol- ogy struc- ture
k	Input↔ : wavenum- ber in unit of 1/Mpc.
z	Input↔ : red- shift

<i>SPLIT</i>	Input : switch to set the method of wiggle- nowiggle split
--------------	---

value of NL IR-resumed power spectrum



5.7.2.5 pm_nowiggle() `double pm_nowiggle (`
 `struct Cosmology * Cx,`
 `double k,`
 `double z,`
 `double kf0,`
 `int cleanup,`
 `long SPLIT)`

Compute the no-wiggle componenet of the matter power spectrum.

Parameters

<i>Cx</i>	Input↔ : pointer to cos- mol- ogy struc- ture
<i>k</i>	Input↔ : wavenum- ber in unit of h/Mpc.
<i>z</i>	Input↔ : red- shift
<i>kf0</i>	Input↔ : first ele- ment of the k- array, used in nor- mal- ization of EH no- wiggle spec- trum

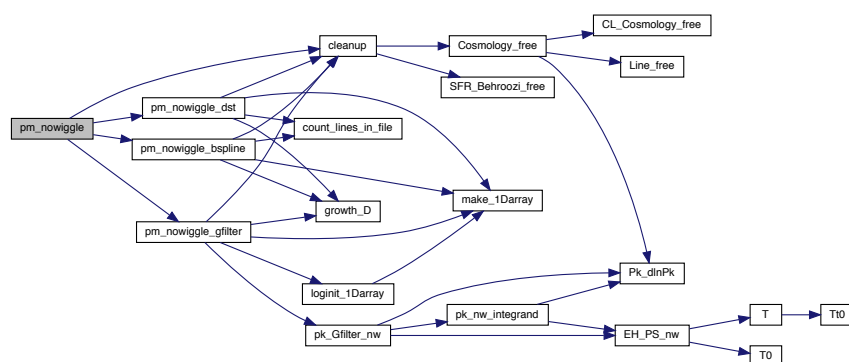
Parameters

<i>cleanup</i>	Input↔ : switch to set whether to free the mem- ory allo- cated to no- wiggle inter- pola- tors
<i>SPLIT</i>	Input↔ : switch to set the method of wiggle- nowiggle split

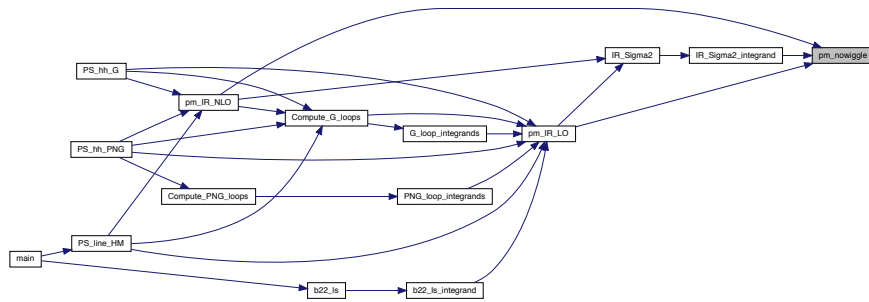
Returns

double value of no-wiggle power spectrum

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.2.6 pm_nowiggle_bspline() `double pm_nowiggle_bspline (`
 `struct Cosmology * Cx,`
 `double k,`
 `double z,`
 `int cleanup)`

Compute the no-wiggle component of the matter power spectrum, reading in and interpolating the output of apython code which computed the broadband by fitting families of Bsplines (see Vlah et al 2015)

Parameters

Cx	Input↔ : pointer to cos- mol- ogy struc- ture
k	Input↔ : wavenum- ber in unit of h/Mpc.
z	Input↔ : red- shift

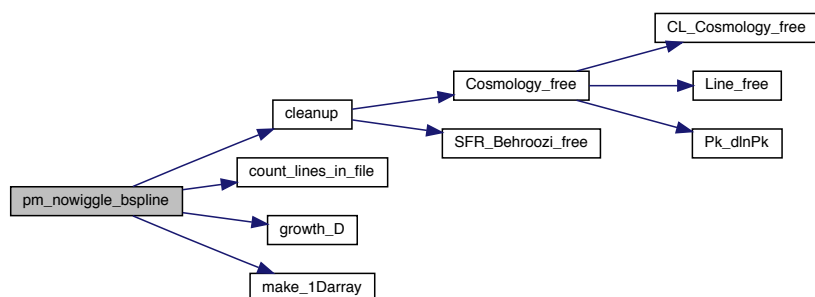
Parameters

<i>cleanup</i>	Input↔ : switch to set whether to free the mem- ory allo- cated to no- wiggle inter- polators
----------------	---

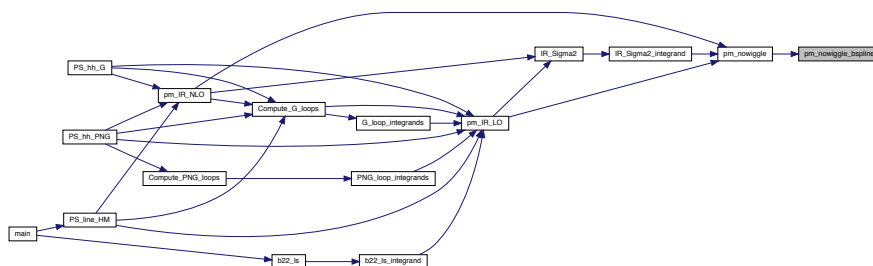
Returns

double value of no-wiggle power spectrum

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.2.7 pm_nowiggle_dst() `double pm_nowiggle_dst (`
 `struct Cosmology * Cx,`
 `double k,`
 `double z,`
 `int cleanup)`

Compute the no-wiggle componenet of the matter power spectrum, reading in and interpolating the output of apython code which computed the broadband by discrete sin-transform, See Hamann et al 2010.

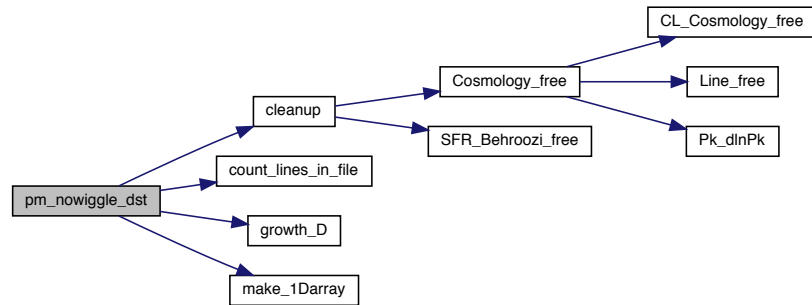
Parameters

<i>Cx</i>	Input↔ : pointer to cos- mol- ogy struc- ture
<i>k</i>	Input↔ : wavenum- ber in unit of h/Mpc.
<i>z</i>	Input↔ : red- shift
<i>cleanup</i>	Input↔ : switch to set whether to free the mem- ory allo- cated to no- wiggle inter- pola- tors

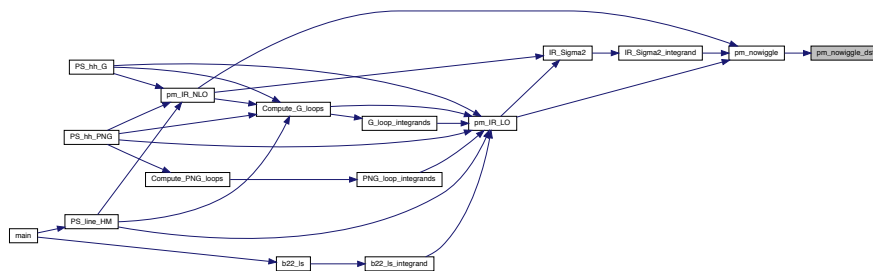
Returns

double value of no-wiggle power spectrum

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.7.2.8 pm_nowiggle_gfilter() double pm_nowiggle_gfilter (
    struct Cosmology * Cx,
    double k,
    double z,
    int cleanup )
  
```

Compute the no-wiggle componenet of the matter power spectrum, using Gaussian filter (see Vlah et al 2015)

Parameters

Cx	Input↔ : pointer to cos- mol- ogy struc- ture
----	--

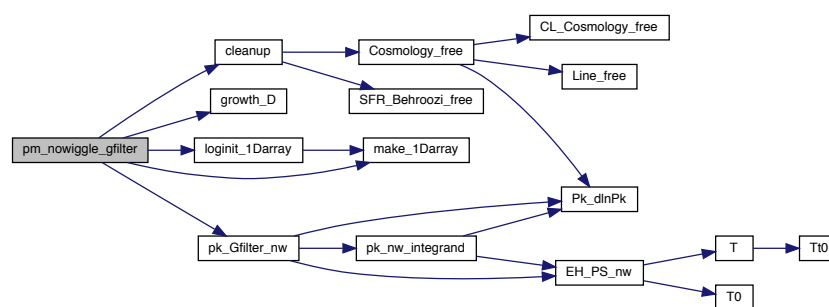
Parameters

k	Input↔ : wavenum- ber in unit of h/Mpc.
z	Input↔ : red- shift
<i>cleanup</i>	Input↔ : switch to set whether to free the mem- ory allo- cated to no- wobble inter- polators

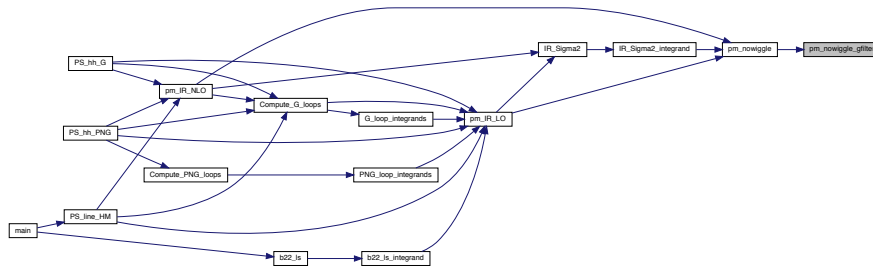
Returns

double value of no-wobble power spectrum

Here is the call graph for this function:



Here is the caller graph for this function:



5.8 line_ingredients.c File Reference

Documented line_ingredients module.

```
#include "header.h"
```

Include dependency graph for line_ingredients.c:



Functions

- struct [Line](#) * [Line_alloc_init](#) (struct [Cosmology](#) *Cx, long line_type, size_t npoints_interp, double M_min, long mode_mf)
Allocate the memory and initialize the the line structure.
- int [Line_free](#) (struct [Line](#) *Lx)
Free the line structure.
- int [Line_evaluate](#) (struct [Line](#) *Lx, double *zz, double *res)
Allocate the memory and initialize the the line structure.
- double [mult_func](#) (double sigma, long mode_mf)
Compute the multiplicity function needed to compute the halo mass function Three models are implemented: Press-Schechter, Sheth-Tormen and Tinker see Pillepich et al arxiv: 0811.4176 for the expressions.
- double [mass_func](#) (struct [Cosmology](#) *Cx, double M, double z, long mode_mf)
Compute the halo mass function for Press-Schechter, Sheth-Tormen and Tinker models see Pillepich et al arxiv: 0811.4176 for the expressions.
- double [mass_func_sims](#) (struct [Cosmology](#) *Cx, double M, double z, long mode_mf)
Read in the measured mass function of Hidden-valey sims and build an interpolator for HMF(M) for a fixed redshift.
- void [halo_bias](#) (struct [Cosmology](#) *Cx, double M, double z, long mode_mf, double *bias_arr)
computes the halo biases for three mass functions, press-schechter, Sheth-Tormen, and Tinker mass functions
- void [logSFR_Behroozi_read](#) (double *z_arr, double *logM_arr, double *log10SFR)
Read in the file for the star formation rate by Behroozi et al 2013.
- int [logSFR_alloc_init](#) ()
Allocate memory and initialize the 2d interpolator for the star formation rate of Behroozi et al 2013 as a function of halo mass and redshift.
- int [SFR_Behroozi_free](#) ()
Free the memory allocated to the interpolators of star formation rate by Behroozi et al 2013.
- double [logSFR_Behroozi](#) (double logM, double z)

Evaluate the SFR interpolator object for a given value of mass and redshift.

- double [luminosity](#) (double M, double z, long mode_lum)
Compute the line specific luminosity in unit of solar luminosity For CO ladder, I am using the fits in Table 4 of ??? et al arXiv:1508.05102, while for CII we use Silva et al arXiv:
- int [mass_moment1_integ](#) (unsigned nd, const double *x, void *p, unsigned fdim, double *fvalue)
Compute the first luminosity-weighted mass moment.
- double [mass_moment1](#) (struct [Cosmology](#) *Cx, double z, double M_min, long mode_mf, long mode_lum)
in unit of $M_{\text{sun}}/\text{Mpc}^3$
- int [mass_moment2_integ](#) (unsigned nd, const double *x, void *p, unsigned fdim, double *fvalue)
Compute the second luminosity-weighted mass moment.
- double [mass_moment2](#) (struct [Cosmology](#) *Cx, double z, double M_min, long mode_mf, long mode_lum)
in unit of $M_{\text{sun}}/\text{Mpc}^3$
- int [bias_lum_weighted_integ](#) (unsigned nd, const double *x, void *p, unsigned fdim, double *fvalue)
Compute the luminosity-weighted linear and quadratic line biases.
- void [bias_lum_weighted](#) (struct [Cosmology](#) *Cx, double z, double M_min, long mode_mf, long mode_lum, double *result)
- double [p_sig_shot_integrand](#) (double x, void *par)
Model from Keating et al 2016 to account for the observed variation in halo activity, i.e.
- double [p_sig_shot](#) (double scatter)
- double [p_sig_Tbar_integrand](#) (double x, void *par)
Model from Keating et al 2016 to account for the observed variation in halo activity, i.e.
- double [p_sig_Tbar](#) (double scatter)
- void [line_bias](#) (struct [Line](#) *Lx, double z, double *result)
Compute the linear and quadratic line biases, accounting for the normalization w.r.t.
- double [mean_intens](#) (struct [Cosmology](#) *Cx, size_t line_id, double z)
Compute the line mean intensity in unit of $\text{erg Mpc}^{-2} \text{Sr}^{-1}$.
- double [Tbar_line](#) (struct [Cosmology](#) *Cx, size_t line_id, double z)
Compute the mean brightness temperature of CO in unit of microK, compared with Pullen et al and Lidz et al 2011.

Variables

- struct [globals](#) [gb](#)

5.8.1 Detailed Description

Documented line_ingredients module.

This module includes functions that are needed for computing the line clustering and shot contributions.

Azadeh Moradinezhad Dizgah, November 4th 2021

In summary, the following functions can be called from other modules:

1. [Line_alloc_init\(\)](#) allocate memory and initialized the line structure which contains 4 interpolators for first and second mass moments and linear and quadratic line biases.
2. [Line_free\(\)](#) frees the memory allocated to line structure
3. [Line_evaluate\(\)](#) evaluates the interpolators initialized in [Line_alloc_init\(\)](#)
4. [mult_func\(\)](#) computes the multiplicity function needed for computing the halo mass function

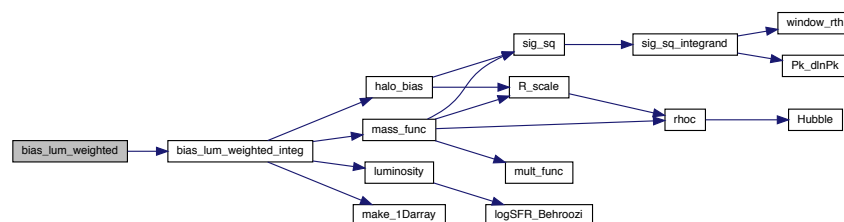
5. `mass_func()` computes the halo mass finction. Three options are available, Press-Schechter, Sheth-Tormen, Tinker
6. `mass_func_sims()` reads in the measured mass function on Hidden-Valley simulations by Farnik, and convert it to compare with the theoretical predictions
7. `halo_bias()` computes the halo biases assuming the above theoretical predictions of the halo mass function
8. `logSFR_Behroozi_read()` reeds in the data file of Behroozi 2013 for SFR(M,z)
9. `logSFR_alloc_init()` allocates memory for 2d interpolator of logSFR(M,z)
10. `SFR_behroozi_free()` frees the memory allocated to logSFR interpolator
11. `logSFR_Behroozi()` evaluates the logSFR_Behroozi interpolator
12. `luminosity()` computes the line luminosity
13. `mass_moment1()` computes the first mass moment
14. `mass_moment2()` computes the first mass moment
15. `bias_lum_weighted()` computes the luminosity-weighetd line bias
16. `p_sig_shot()` computes the coefficient accounting for the scatter in L(M) in shot noise
17. `p_sig_Tbar()` computes the coefficient accounting for the scatter in L(M) in mean brightness temprature
18. `mean_intens()` compues the mean intensity of the line
19. `Tbar_line()` compues the mean brightness temprature of the line

5.8.2 Function Documentation

5.8.2.1 bias_lum_weighted() `void bias_lum_weighted (`
`struct Cosmology * Cx,`
`double z,`
`double M_min,`
`long mode_mf,`
`long mode_lum,`
`double * result)`

In units of solar mass;

In units of solar massHere is the call graph for this function:



Here is the caller graph for this function:



5.8.2.2 bias_lum_weighted_integ() `int bias_lum_weighted_integ (`
 unsigned *nd*,
 const double * *x*,
 void * *p*,
 unsigned *fdim*,
 double * *fvalue*)

Compute the luminosity-weighted linear and quadratic line biases.

The normalization of first mass moment is not included yet. The function [bias_lum_weighted_integ\(\)](#) is the integrand and [bias_lum_weighted\(\)](#) computes the bias

Parameters

<i>Cx</i>	Input↔ : pointer to cos- mol- ogy struc- ture
<i>z</i>	Input↔ : red- shift
<i>M_min</i>	Input↔ : min- imum halo mass
<i>mode_mf</i>	Input↔ : model of halo mass func- tion to con- sider, PSC, ST, TR

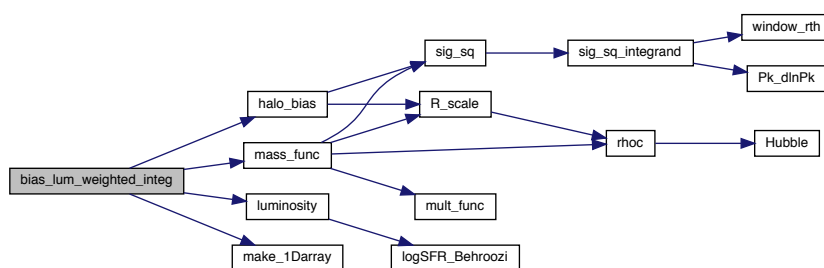
Parameters

<i>mode_lum</i>	Input↔ : which lumi- nosity model, basi- cally which line con- sid- ered
-----------------	---

Returns

un-normalized line bias

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.3 halo_bias() `void halo_bias (`
 `struct Cosmology * Cx,`
 `double M,`
 `double z,`
 `long mode_mf,`
 `double * bias_arr)`

computes the halo biases for three mass functions, press-schechter, Sheth-Tormen, and Tinker mass functions

Parameters

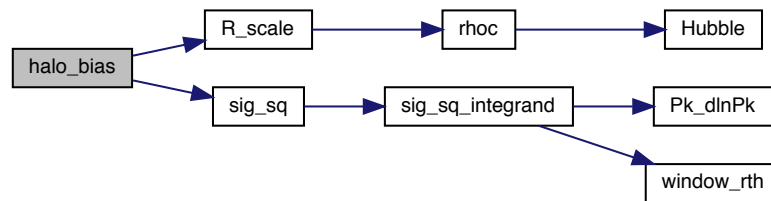
C_x	Input↔ : Cosmology struc- ture
M	Input↔ : halo mass
z	Input↔ : red- shift
$mode_mf$	Input↔ : switch for setting the model of mass func- tion, can be set to PSC, ST, TR
$bias_arr$	Output↔ : the output array con- tain- ing linear and quadratic local- in- matter halo biases, and quadratic and cubic tidal biases

Returns

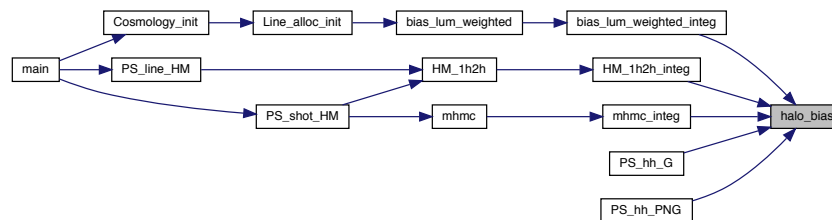
void

Note that for PSC and ST mass functions, same form of the biases can be assumed, with different coefficients. See [astro-ph/0006319](#)

Assuming spherical collapseHere is the call graph for this function:



Here is the caller graph for this function:



```

5.8.2.4 Line_alloc_init() struct Line * Line_alloc_init (
    struct Cosmology * Cx,
    long line_type,
    size_t npoints_interp,
    double M_min,
    long mode_mf )

```

Allocate the memory and initialize the the line structure.

This structure contains interpolators for computing the luminosity-weighted mass moments and line biases For a given line defined with "line_type" variable, this function first computes the above four quantities for a wide range of redshifts. Next it iniialized 4 interpolators for these quantities, and store them in line structure.

Parameters

Cx	Input↔
	:
	Cosmology
	struc-
	ture

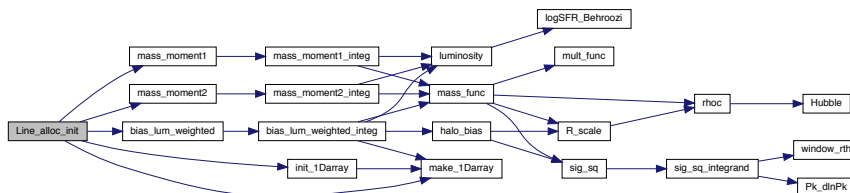
Parameters

<i>line_type</i>	Input↔ : name of the line to compute. It can be set to CII, CO10, CO21, CO32, CO43, CO54, CO65
<i>npoints_interp</i>	Input↔ : number of interpolation points
<i>M_min</i>	Input↔ : minimum halo mass for mass integrals
<i>mode_mf</i>	Input↔ : theoretical model of halo mass function to use. It can be set to sheth-↔ Tormen (ST), Tinker (TR) or Press-↔ Schechter (PSC)

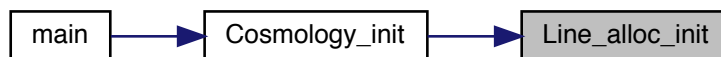
Returns

the total clustering line power spectrum, including the 1- and 2-halo term

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.8.2.5 line_bias() void line_bias (
    struct Line * Lx,
    double z,
    double * result )
  
```

Compute the linear and quadratic line biases, accounting for the normalization w.r.t.

the first mass moment

Parameters

<i>Lx</i>	Input↔ : Pointer to line struc- ture
<i>z</i>	Input↔ : Red- shift

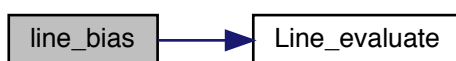
Parameters

<i>result</i>	Input↔ : a pointer to an array con- taining the re- sults of b1↔ _line and b2↔ line
---------------	--

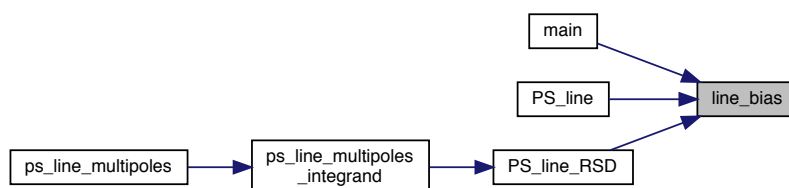
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.8.2.6 Line_evaluate()  int Line_evaluate (
    struct Line * Lx,
    double * zz,
    double * res )
```

Allocate the memory and initialize the the line structure.

This structure contains interpolators for computing the luminosity-weighted mass moments and line biases For a given line defined with "line_type" variable, this function first computes the above four quantities for a wide range of redshifts. Next it iniialized 4 interpolators for these quantities, and store them in line structure.

Parameters

<i>Lx</i>	Input↔ : Pointer to the line struc- ture
-----------	--

Parameters

zz	<p>Input↵ : this is an array with 4 ele- ments to deter- mine which of the 4 in- terpo- lators should be evalu- ated.</p> <ul style="list-style-type: none">• If any of the el- e- ments are set to DO↵ ↵ ↵ NOT↵ ↵ ↵ EVALUATE, the quan- ti- tiy cor- re- spond- ing to that in- dex is not com- puted. O• If any of
	<p>the el- e- ments</p>

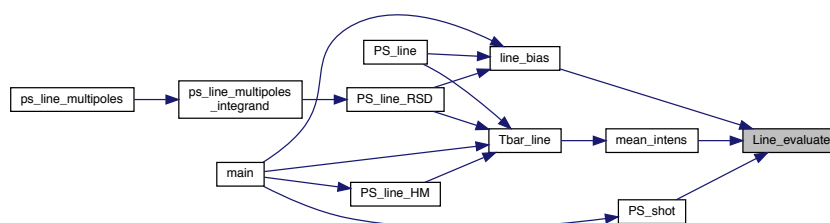
Parameters

<i>res</i>	Output : an array containing the results. The number of elements of this array depends on how the <code>zz</code> array is set.
------------	--

Returns

the error status

Here is the caller graph for this function:

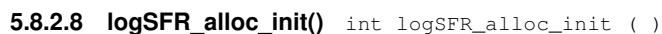


5.8.2.7 Line_free() `int Line_free (`
`struct Line * Lx)`

Free the line structure.

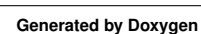
Returns

Here is the caller graph for this function:

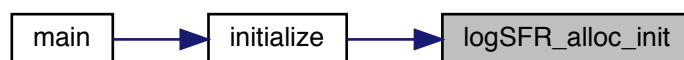


Returns

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.9 logSFR_Behroozi() `double logSFR_Behroozi (`
 `double logM,`
 `double z)`

Evaluate the SFR interpolator object for a given value of mass and redshift.

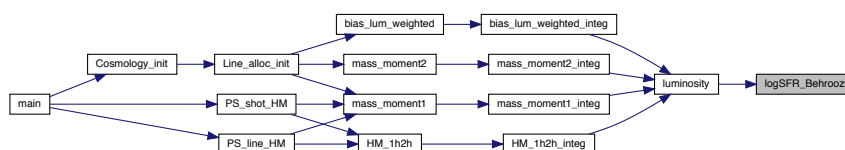
Parameters

<i>logM</i>	Input↔ : log10 of halo mass
<i>z</i>	Input↔ : red- shift

Returns

log10SFR

Here is the caller graph for this function:



5.8.2.10 logSFR_Behroozi_read() `void logSFR_Behroozi_read (`
 `double * z_arr,`
 `double * logM_arr,`
 `double * log10SFR)`

Read in the file for the star formation rate by Behroozi et al 2013.

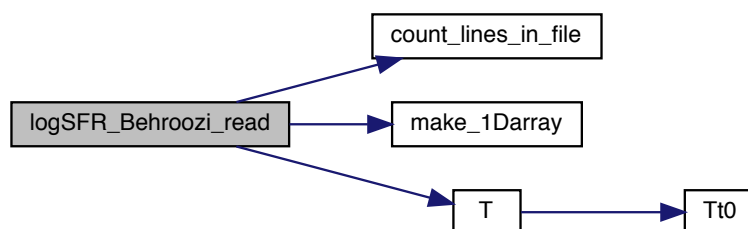
Parameters

<i>z_arr</i>	Output : pointer to an array of red- shifts read from the file
<i>logM_arr</i>	Output : pointer to an array of halo masses read from the file
<i>log10SFR</i>	Output : pointer to an array of SFR read from the file

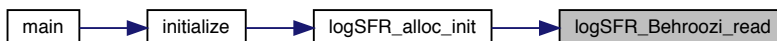
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.11 luminosity() `double luminosity (`
 `double M,`
 `double z,`
 `long mode_lum)`

Compute the line specific luminosity in unit of solar luminosity For CO ladder, I am using the fits in Table 4 of ??? et al arXiv:1508.05102, while for CII we use Silva et al arXiv:

Parameters

<i>M</i>	Input↵ : halo mass
<i>z</i>	Input↵ : red- shift
<i>mode_lum</i>	Input↵ : which lumi- nosity model, basi- cally which line con- sid- ered

Returns

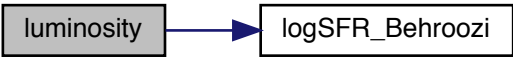
line luminosity

a = 1.37 Charilli

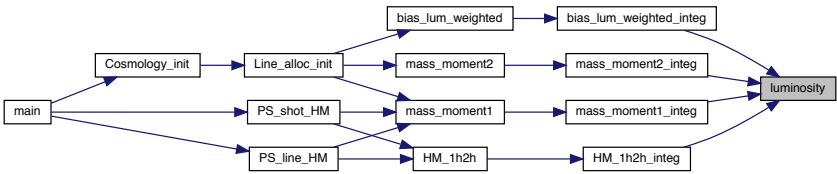
b = -1.74

in unit of K km/s pc²

in unit of L_{sun} Here is the call graph for this function:



Here is the caller graph for this function:



```
5.8.2.12 mass_func() double mass_func (
    struct Cosmology * Cx,
    double M,
    double z,
    long mode_mf )
```

Compute the halo mass function for Press-Schechter, Sheth-Tormen and Tinker models see Pillepich et al arxiv: 0811.4176 for the expressions.

Parameters

Cx	Input↔ : Cosmology struc- ture
M	Input↔ : Halo mass func- tion
z	Input↔ : red- shift

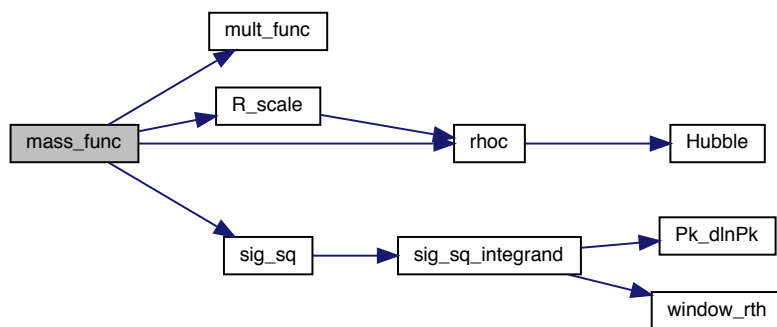
Parameters

<i>mode_mf</i>	Input↔ : switch for setting the model of mass func- tion, can be set to PSC, ST, TR
----------------	---

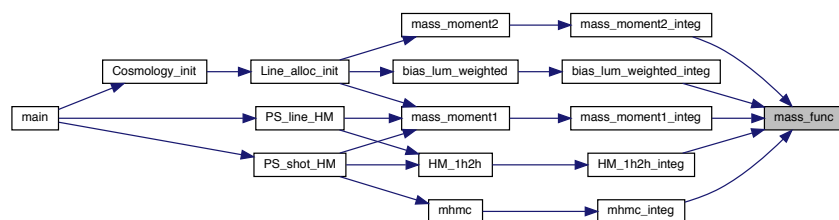
Returns

the halo mass function in unit of halos per Mpc^3 per solar mass, compared at $z=0$ with Murray etal <https://arxiv.org/abs/1306.5140>

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.8.2.13 mass_func_sims() double mass_func_sims (
    struct Cosmology * Cx,
    double M,
    double z,
    long mode_mf )

```

Read in the measured mass function of Hidden-valey sims and build an interpolator for HMF(M) for a fixed redshift.

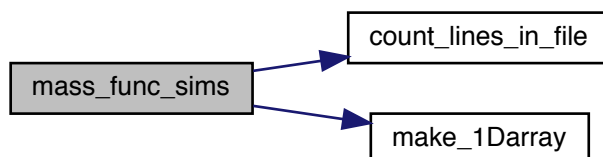
Parameters

<i>Cx</i>	Input↔ : <i>Cosmology</i> struc- ture
<i>M</i>	Input↔ : halo mass
<i>z</i>	Input↔ : red- shift
<i>mode_mf</i>	Input↔ : switch for setting the model of mass func- tion, can be set to PSC, ST, TR

Returns

the interpolated measured halo mass function
M in unit of M_{sun} and HMF in unit of $\# \text{-of-halos} / \text{Mpc}^3 / M_{\text{sun}}$

Here is the call graph for this function:

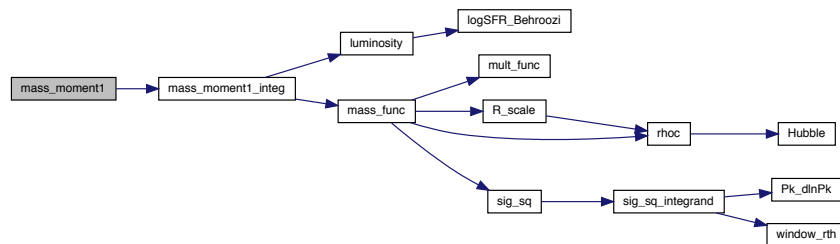


5.8.2.14 mass_moment1() `double mass_moment1 (`
`struct Cosmology * Cx,`
`double z,`
`double M_{min} ,`
`long mode_mf,`
`long mode_lum)`

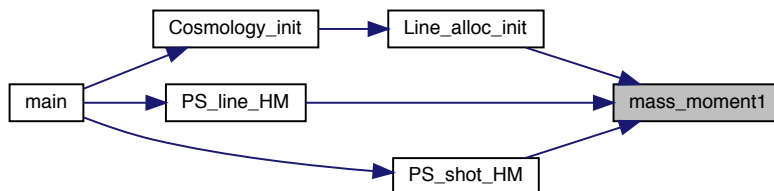
in unit of $M_{\text{sun}}/\text{Mpc}^3$

In units of solar mass;

In units of solar mass Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.15 mass_moment1_integ() `int mass_moment1_integ (`
`unsigned nd,`
`const double * x,`
`void * p,`
`unsigned fdim,`
`double * fvalue)`

Compute the first luminosity-weighted mass moment.

The function `mass_moment1_integ()` is the integrand and `mass_moment1()` compute the moment

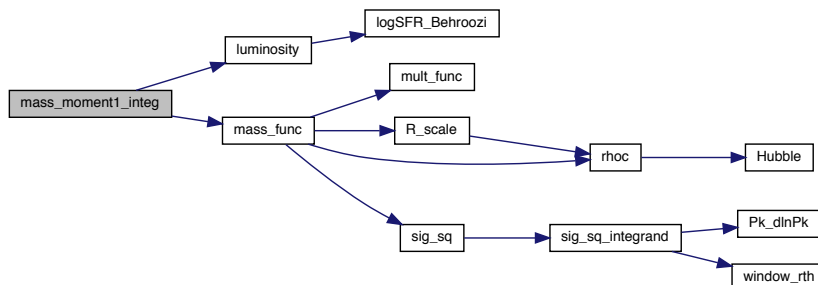
Parameters

C_x	Input↔ : pointer to cos- mol- ogy struc- ture
z	Input↔ : red- shift
M_{min}	Input↔ : min- imum halo mass
$mode_{mf}$	Input↔ : model of halo mass func- tion to con- sider, PSC, ST, TR
$mode_{lum}$	Input↔ : which lumi- nosity model, basi- cally which line con- sid- ered

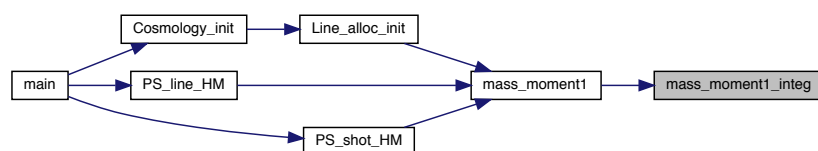
Returns

the first mass moment

Here is the call graph for this function:



Here is the caller graph for this function:

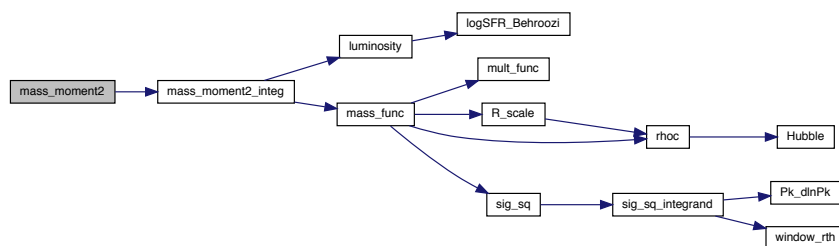


5.8.2.16 mass_moment2() double mass_moment2 (
 struct `Cosmology` * `Cx`,
 double `z`,
 double `M_min`,
 long `mode_mf`,
 long `mode_lum`)

in unit of $M_{\text{sun}}/\text{Mpc}^3$

In units of solar mass;

In units of solar massHere is the call graph for this function:



Here is the caller graph for this function:



5.8.2.17 mass_moment2_integ() `int mass_moment2_integ (`
 `unsigned nd,`
 `const double * x,`
 `void * p,`
 `unsigned fdim,`
 `double * fvalue)`

Compute the second luminosity-weighted mass moment.

The function [mass_moment2_integ\(\)](#) is the integrand and [mass_moment2\(\)](#) compute the moment

Parameters

<i>Cx</i>	Input↔ : pointer to cos- mol- ogy struc- ture
<i>z</i>	Input↔ : red- shift
<i>M_min</i>	Input↔ : min- imum halo mass
<i>mode_mf</i>	Input↔ : model of halo mass func- tion to con- sider, PSC, ST, TR

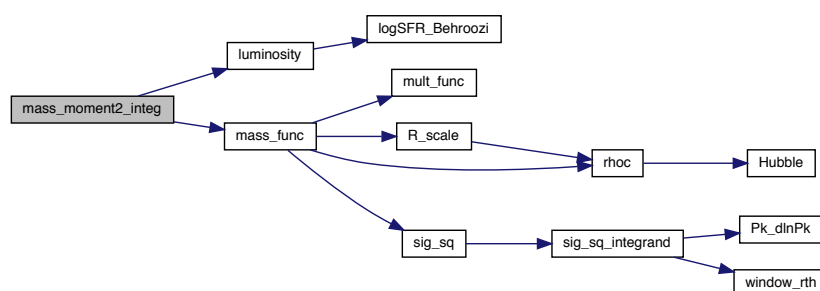
Parameters

<i>mode_lum</i>	Input↔ : which lumi- nosity model, basi- cally which line con- sid- ered
-----------------	---

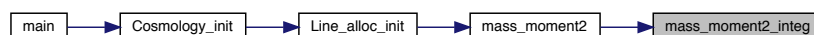
Returns

the second mass moment

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.8.2.18 mean_intens() double mean_intens (
    struct Cosmology * Cx,
    size_t line_id,
    double z )
  
```

Compute the line mean intensity in unit of $\text{erg Mpc}^{-2} \text{Sr}^{-1}$.

Parameters

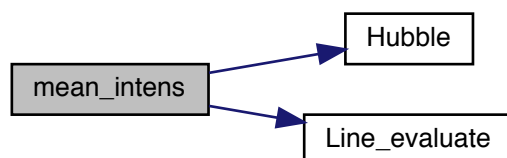
<i>Cx</i>	Input↔ : Pointer to cos- mol- ogy struc- ture
<i>line_id</i>	Input↔ : id of line of inter- est, an integer value
<i>z</i>	Input↔ : Red- shift

Returns

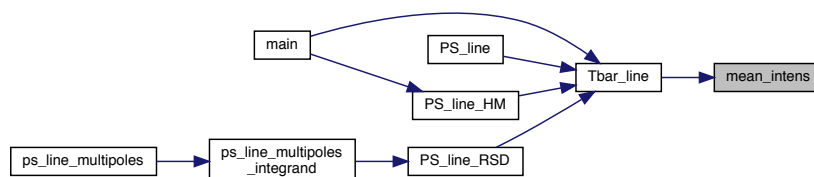
the line mean intensity

Note: ν_J is the rest-frame emission frequency related to the observed frequency as $\nu_{\text{obs}} = \nu_J / (1+z_J)$ For a CO transition from $J \rightarrow J-1$, the rest-frame frequency is $\nu_J = J \nu_{\text{CO}}$ where $\nu_{\text{CO}} = 115 \text{ GHz}$.

in unit of erg/sHere is the call graph for this function:



Here is the caller graph for this function:



5.8.2.19 mult_func() `double mult_func (`
`double sigma,`
`long mode_mf)`

Compute the multiplicity function needed to compute the halo mass function Three models are implemented: Press-Schechter, Sheth-Tormen and Tinker see Pillepich et al arxiv: 0811.4176 for the expressions.

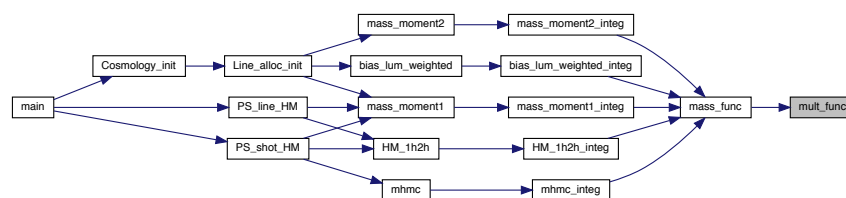
Parameters

<i>sigma</i>	Input↔ : variance of matter fluctuations
<i>mode_mf</i>	Input↔ : switch for setting the model of mass function, can be set to PSC, ST, TR

Returns

the multiplicity function

In Barkana & Loeb Rev a = 0.75 Here is the caller graph for this function:

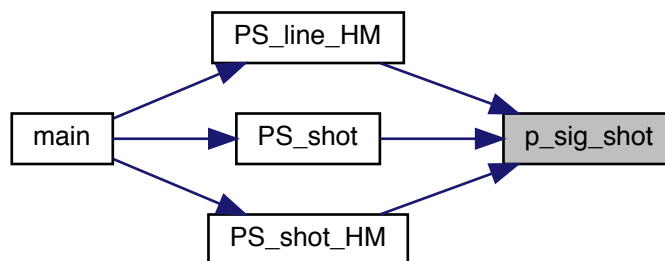


5.8.2.20 p_sig_shot() `double p_sig_shot (`
`double scatter)`

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.21 p_sig_shot_integrand() `double p_sig_shot_integrand (`
 `double x,`
 `void * par)`

Model from Keating et al 2016 to account for the observed variation in halo activity, i.e.

scatter in the L(M) relation `p_sig_shot` replaces the `f_duty` in the shot-noise used in some LIM paper (ex. Lidz et al 2011). `p_sig_shot_integrand()` is the integrand, and `p_sig_shot()` computes the scatter factor for the shot noise.

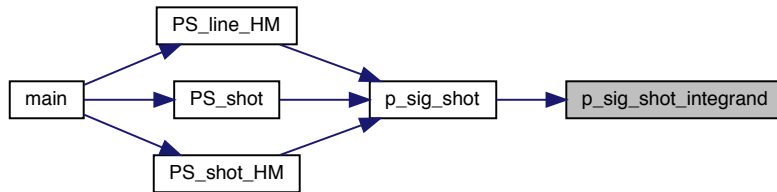
Parameters

<i>scatter</i>	Input↔ : variance of the log- scatter
----------------	---

Returns

the scatter coeff of the shot noise

Here is the caller graph for this function:

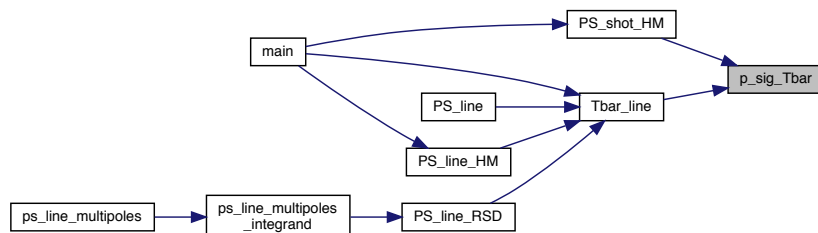


5.8.2.22 `p_sig_Tbar()` `double p_sig_Tbar (`
`double scatter)`

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.23 `p_sig_Tbar_integrand()` `double p_sig_Tbar_integrand (`
`double x,`
`void * par)`

Model from Keating et al 2016 to account for the observed variation in halo activity, i.e.

scatter in the L(M) relation `p_sig_Tbar` replace the `f_duty` in the average brightness temprature used in some LIM paper (ex. Lidz et al 2011). `p_sig_Tbar_integrand()` is the integrand, and `p_sig_Tbar()` computes the scatter factor for the mean brightness temprature.

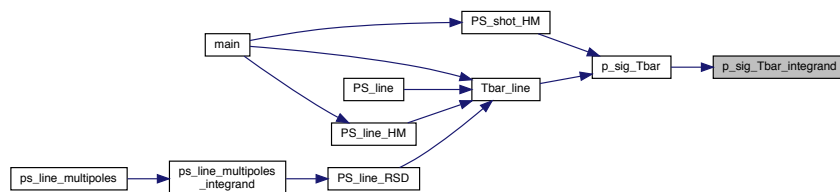
Parameters

scatter	Input↔ : variance of the log- scatter
----------------	---

Returns

the scatter coeff of Tbar

Here is the caller graph for this function:

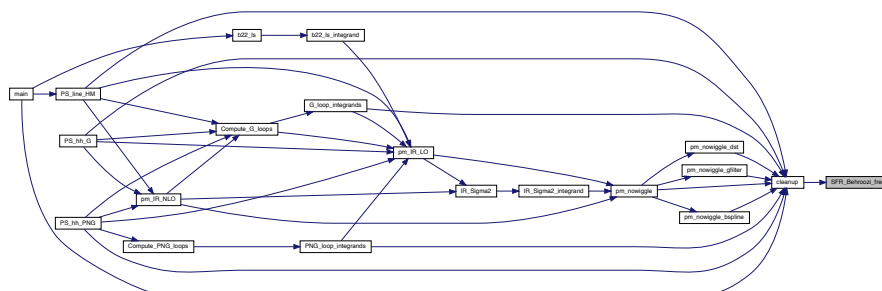
**5.8.2.24 SFR_Behroozi_free()** `int SFR_Behroozi_free ()`

Free the memory allocated to the interpolators of star formation rate by Behroozi et al 2013.

Returns

the error status

Here is the caller graph for this function:

**5.8.2.25 Tbar_line()** `double Tbar_line (`
`struct Cosmology * Cx,`
`size_t line_id,`
`double z)`

Compute the mean brightness temprature of CO in unit of microK, compared with Pullen et al and Lidz et al 2011.

Parameters

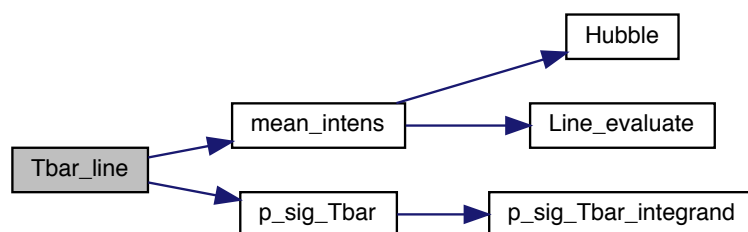
Cx	Input↔ : Pointer to cos- mol- ogy struc- ture
line_id	Input↔ : id of line of inter- est, an integer value
z	Input↔ : Red- shift

Returns

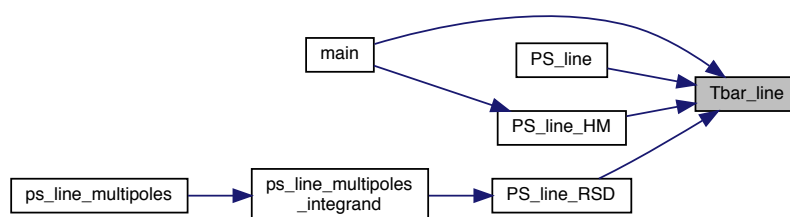
the line mean temprature assuming Rayleigh-Jeans limit

Boltzmann constant in unit of erg K⁻¹

factor of 10⁶ is the conversion factor from K to microKHere is the call graph for this function:



Here is the caller graph for this function:



5.8.3 Variable Documentation

5.8.3.1 gb struct globals gb

5.9 main.c File Reference

Documented main module, including functions to initialize and cleanup the cosmology structure and examples of calls to functions in other modules to compute the line clustering and shot power spectrum.

```
#include "header.h"
```

Include dependency graph for main.c:



Functions

- int [main](#) (int argc, char *argv[])
- void [initialize](#) ()

Initilize the path to the required directories, set the values of cosmological parmaeters, and initialize the interpolator of the SFR(M,z) from tabulated data provided in gb.SFR_filename.
- void [cleanup](#) (struct [Cosmology](#) *Cx)

Free the memory allocated to cosmology structure and SFR interpolator.

Variables

- struct [globals](#) gb

5.9.1 Detailed Description

Documented main module, including functions to initialize and cleanup the cosmology structure and examples of calls to functions in other modules to compute the line clustering and shot power spectrum.

Azadeh Moradinezhad Dizgah, November 4th 2021

In order to call any function from the package, the function calls should be placed in the marked section of [main\(\)](#) function.

5.9.2 Function Documentation

Returns

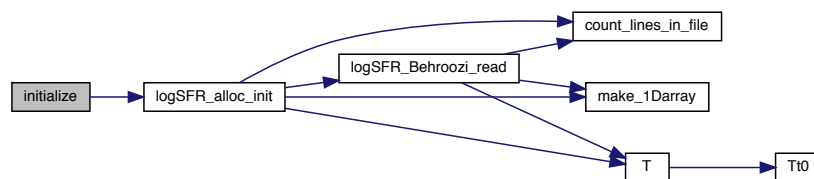
void

Change the path to the parent directory

In units of km/s

$\omega_b = \Omega_b h^2$;

3.0665 Here is the call graph for this function:



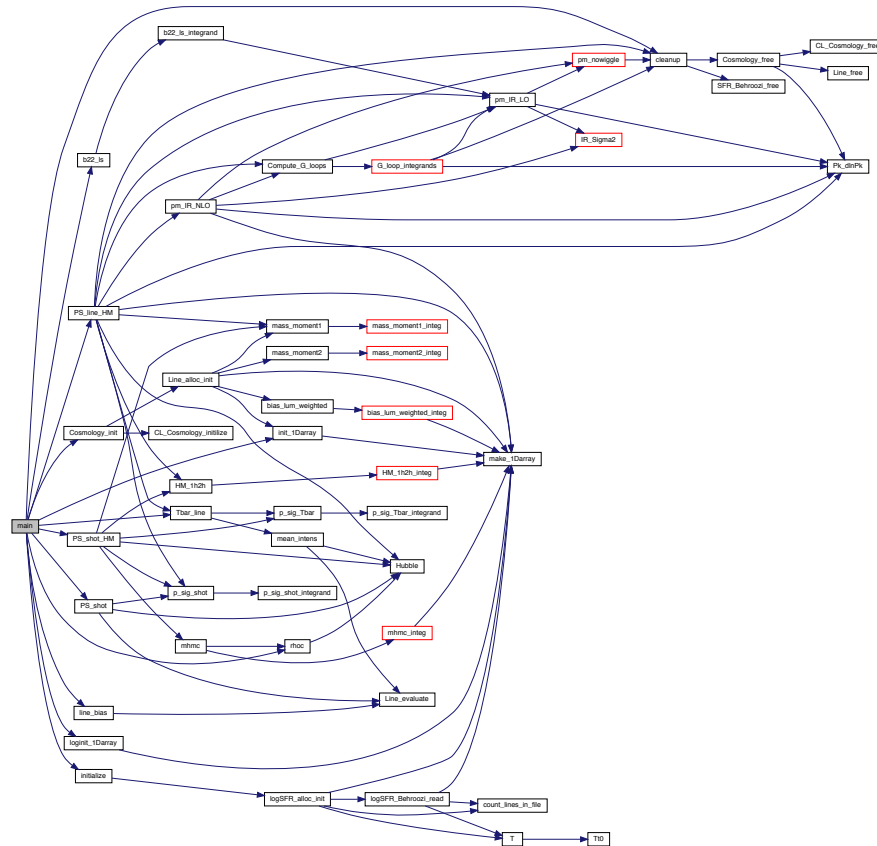
Here is the caller graph for this function:



```

5.9.2.3 main() int main (
    int argc,
    char * argv[] )
  
```


Here is the call graph for this function:



5.9.3 Variable Documentation

5.9.3.1 gb struct globals gb

5.10 ps_halo_1loop.c File Reference

Documented real-space, direct integration computation of 1loop contributions of the halo/galaxy power spectrum
See arXiv:2010.14523 for explicit expressions.

```
#include "header.h"
```

Include dependency graph for ps_halo_1loop.c:



Functions

- double [PS_hh_G](#) (struct [Cosmology](#) *Cx, double k, double z, double M, long mode_pt, long IR_switch, long SPLIT, long mode_mf)
Compute the contributions up to 1loop to halo power spectrum for Gaussian initial conditions.
- double [PS_hh_PNG](#) (struct [Cosmology](#) *Cx, double k, double z, double M, long mode_pt, long IR_switch, long SPLIT, long mode_mf)
Compute contributions up to 1loop to halo power spectrum arising from non-Gaussian initial conditions of local shape.
- void [Compute_G_loops](#) (struct [Cosmology](#) *Cx, double k, double z, long IR_switch, long hm_switch, long SPLIT, double *result)
Compute the loop contributions due to nonlinear evolution of matter fluctuations and nonlinear halo bias, present for Gaussian initial conditions The function [G_loop_integrands\(\)](#) defines the integrand and [Compute_G_loops\(\)](#) computes the integrals.
- static int [G_loop_integrands](#) (const int *ndim, const cubereal x[], const int *ncomp, cubereal ff[], void *p)
- void [Compute_PNG_loops](#) (struct [Cosmology](#) *Cx, double k, double z, long IR_switch, long SPLIT, double *result)
Compute the loop contributions due to nonlinear evolution of matter fluctuations and nonlinear halo bias, rising from non-Gaussian initial conditions of local shape The function [PNG_loop_integrands\(\)](#) defines the integrand and [Compute_PNG_loops\(\)](#) computes the integrals.
- static int [PNG_loop_integrands](#) (const int *ndim, const cubereal x[], const int *ncomp, cubereal ff[], void *p)
- double [F2_s](#) (double k1, double k2, double mu)
- double [S2_s](#) (double k1, double k2, double mu)
- double [F3_s](#) (double k, double q, double mu)
- double [S2](#) (double mu)
- double [F2](#) (double k1, double k2, double mu)

Variables

- struct [globals](#) [gb](#)

5.10.1 Detailed Description

Documented real-space, direct integration computation of 1loop contributions of the halo/galaxy power spectrum See arXiv:2010.14523 for explicit expressions.

Azadeh Moradinezhad Dizgah, November 4th 2021

This module computes the 1loop halo/galaxy power spectrum in real-space via direct numerical integration. IR-resummation and EFT counter terms are included. In addition to loops due to gravitational loops, terms arising only in the presence of local PNG are also included. The explicit expressions of all the loops are given in 2010.14523.

In summary, the following functions can be called from other modules:

1. [PS_hh_G\(\)](#)
2. [PS_hh_PNG\(\)](#)
3. [Compute_Gloops\(\)](#)
4. [Compute_PNGloops\(\)](#)
5. [F2_s\(\)](#)
6. [F3_s\(\)](#)
7. [S2_s\(\)](#)
8. [F2\(\)](#)
9. [S2\(\)](#)

5.10.2 Function Documentation

5.10.2.1 Compute_G_loops() void Compute_G_loops (

```

    struct Cosmology * Cx,
    double k,
    double z,
    long IR_switch,
    long hm_switch,
    long SPLIT,
    double * result )

```

Compute the loop contributions due to nonlinear evolution of matter fluctuations and nonlinear halo bias, present for Gaussian initial conditions The function [G_loop_integrands\(\)](#) defines the integrand and [Compute_G_loops\(\)](#) computes the integrals.

Parameters

<i>Cx</i>	Input↔ : pointer to cos- mol- ogy struc- ture
<i>k</i>	Input↔ : wavenum- ber
<i>z</i>	Input↔ : red- shift of inter- est
<i>M</i>	Input↔ : halo mass, used in com- puting the halo bias
<i>IR_switch</i>	Input↔ : switch to de- cide whether to per- form IR resum- mation or no

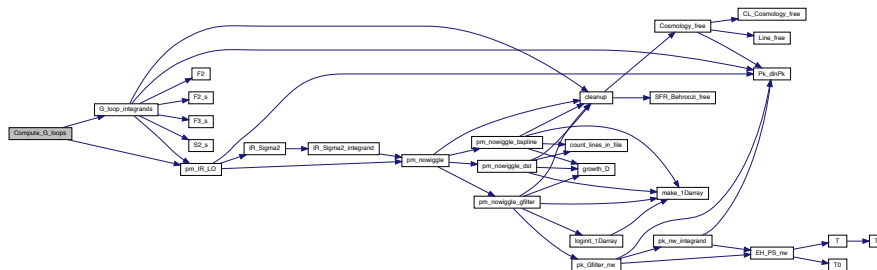
Parameters

<i>hm_switch</i>	Input↔ : switch to de- cide whether to com- pute the 1loop terms due to matter or bias
<i>SPLIT</i>	Input↔ : switch to set the method to per- form the wiggle- nowiggle split of matter power spec- trum
<i>result</i>	Output↔ : an output array con- taining the results of the 1loop terms, has 2 ele- ments for hm_↔ switch=MATTER, and 6 ele- ments for hm_↔ switch=HALO

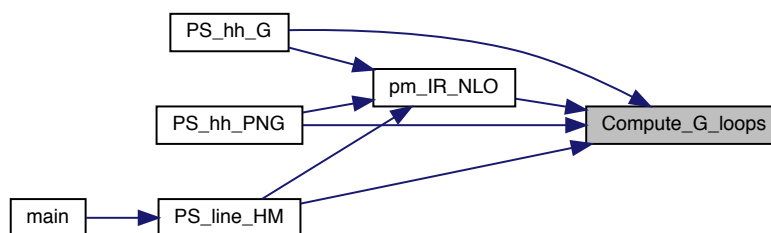
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.2 Compute_PNG_loops()

```
struct Cosmology * Cx,  
double k,  
double z,  
long IR_switch,  
long SPLIT,  
double * result )
```

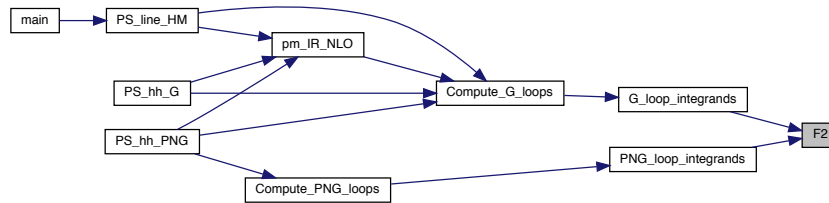
Compute the loop contributions due to nonlinear evolution of matter fluctuations and nonlinear halo bias, arising from non-Gaussian initial conditions of local shape. The function `PNG_loop_integrands()` defines the integrand and `Compute PNG loops()` computes the integrals.

Parameters

C_x	Input↔ : pointer to cos- mol- ogy struc- ture
k	Input↔ : wavenum- ber
z	Input↔ : red- shift of inter- est
<i>IR_switch</i>	Input↔ : switch to de- cide whether to per- form IR resum- mation or no
<i>SPLIT</i>	Input↔ : switch to set the method to per- form the wiggles- nowiggles split of matter power spec- trum

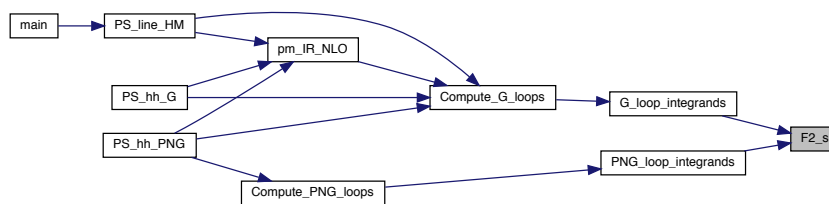
5.10.2.3 F2() `double F2 (`
`double k1,`
`double k2,`
`double mu)`

Here is the caller graph for this function:



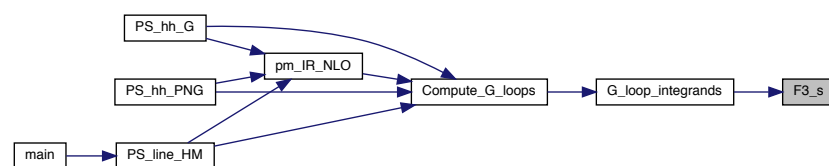
5.10.2.4 F2_s() `double F2_s (`
`double k1,`
`double k2,`
`double mu)`

Here is the caller graph for this function:



5.10.2.5 F3_s() `double F3_s (`
`double k,`
`double q,`
`double mu)`

Here is the caller graph for this function:



5.10.2.6 G_loop_integrands() static int G_loop_integrands (

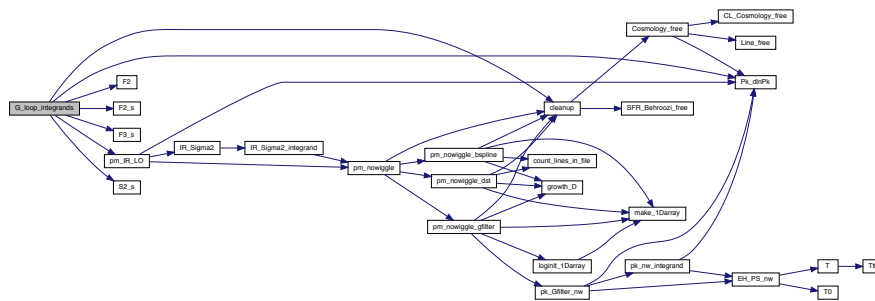
```

const int * ndim,
const cubareal x[],
const int * ncomp,
cubareal ff[],
void * p ) [static]

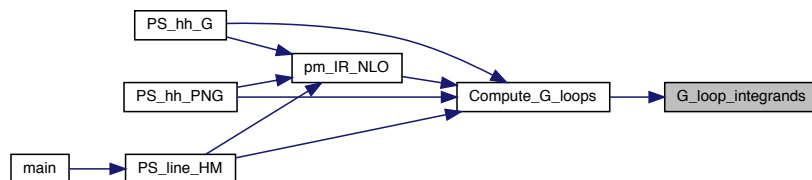
```

Model used in 1907.06666, the integrals are given in the appendix, Eq. A1, note that my $S2_s = \sigma^2(q, k-1)$ and $F2_s = F2(q, k-q)$ in their notation. Factor of $2 \cdot (\log q_{\max} - \log q_{\min})$ is due to change of variable from 0 to logarithmic k , and a factor of 2π is due to integration over azimuthal angle. Note that to compare the theoretical predictions against Emiliano's measurement, since he is using a different notation for Fourier transform, I need to divide each 0 power spectrum by a factor of $1/\text{pow}(2 \cdot M_{\text{PI}}, 3)$, which I do in my `pk_lin()` function. If using another notation for Fourier transform (the one that I usually use, which has a factor of $1/\text{pow}(2 \cdot M_{\text{PI}}, 3)$ in the definition), you need to multiply these integrands by a factor of $1/\text{pow}(2 \cdot M_{\text{PI}}, 3)$.

The integrands below correspond to the following bias combinations: Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.7 PNG_loop_integrands() static int PNG_loop_integrands (

```

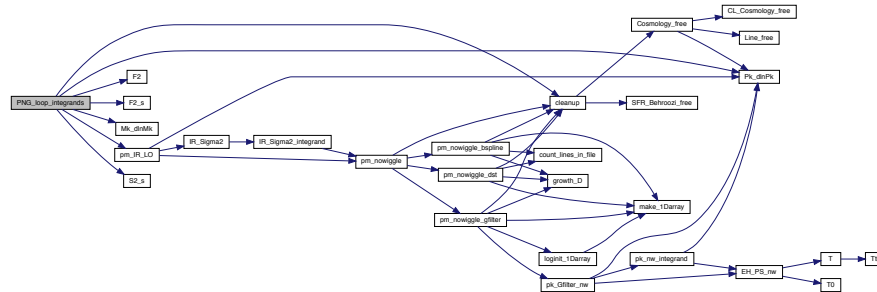
const int * ndim,
const cubareal x[],
const int * ncomp,
cubareal ff[],
void * p ) [static]

```

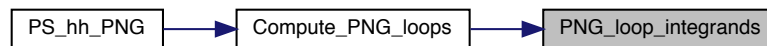
Factor of $2 \cdot (\log q_{\max} - \log q_{\min})$ is due to change of variable from 0 to logarithmic k , and a factor of 2π is due to integration over azimuthal angle. Note that to compare the theoretical predictions against Emiliano's measurement, since he is using a different notation for Fourier transform, I need to divide each 0 power spectrum by a factor of

$1/\text{pow}(2*M_PI,3)$), which I do in my `pk_lin()` function. If using another notation for Fourier transform (the one that I usually use, which has a factor of $1/\text{pow}(2*M_PI,3)$ in the definition), you need to multiply these integrands by a factor of $1/\text{pow}(2*M_PI,3)$.

The integrands below correspond to the following bias combinations: Here is the call graph for this function:



Here is the caller graph for this function:



```
5.10.2.8 PS_hh_G() double PS_hh_G (
    struct Cosmology * Cx,
    double k,
    double z,
    double M,
    long mode_pt,
    long IR_switch,
    long SPLIT,
    long mode_mf )
```

Compute the contributions up to 1loop to halo power spectrum for Gaussian initial conditions.

Parameters

<code>Cx</code>	Input↔ : pointer to cos- mol- ogy struc- ture
<code>k</code>	Input↔ : wavenum- ber

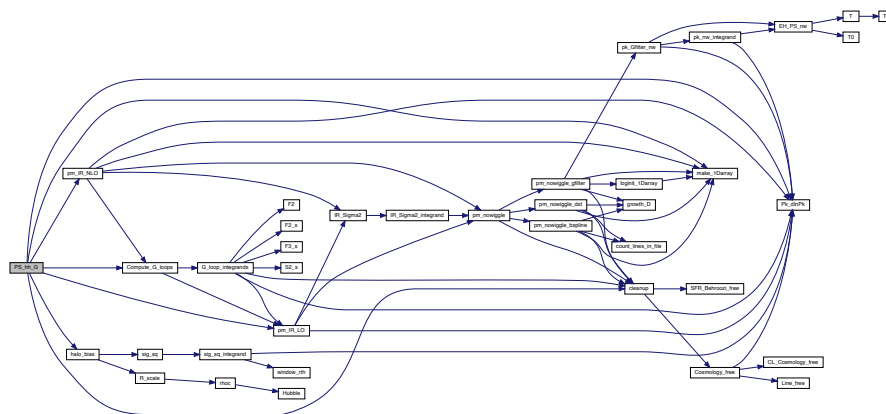
Parameters

<i>z</i>	Input↔ : red-shift of interest
<i>M</i>	Input↔ : halo mass, used in computing the halo bias
<i>mode_pt</i>	Input↔ : switch to decide whether to compute tree-level halo power spectrum or the 1loop
<i>IR_switch</i>	Input↔ : switch to decide whether to perform IR resummation or no

<i>SPLIT</i>	Input↔ : switch to set the method to perform the wiggle-nowiggle split of matter power spectrum
<i>mode_mf</i>	Input↔ : switch to set the theoretical model of the mass function used to compute the halo biases

G loop contributions of P_h

Here is the call graph for this function:



5.10.2.9 PS_hh_PNG() `double PS_hh_PNG (`
 `struct Cosmology * Cx,`
 `double k,`
 `double z,`
 `double M,`
 `long mode_pt,`
 `long IR_switch,`
 `long SPLIT,`
 `long mode_mf)`

Compute contributions up to 1loop to halo power spectrum arising from non-Gaussian initial conditions of local shape.

Parameters

<i>Cx</i>	Input↔ : pointer to cos- mol- ogy struc- ture
<i>k</i>	Input↔ : wavenum- ber
<i>z</i>	Input↔ : red- shift of inter- est
<i>M</i>	Input↔ : halo mass, used in com- puting the halo bias

Parameters

<i>mode_pt</i>	Input↔ : switch to de- cide whether to com- pute tree- level halo power spec- trum or the 1loop
<i>IR_switch</i>	Input↔ : switch to de- cide whether to per- form IR resum- mation or no
<i>SPLIT</i>	Input↔ : switch to set the method to per- form the wiggles- nowiggles split of matter power spec- trum

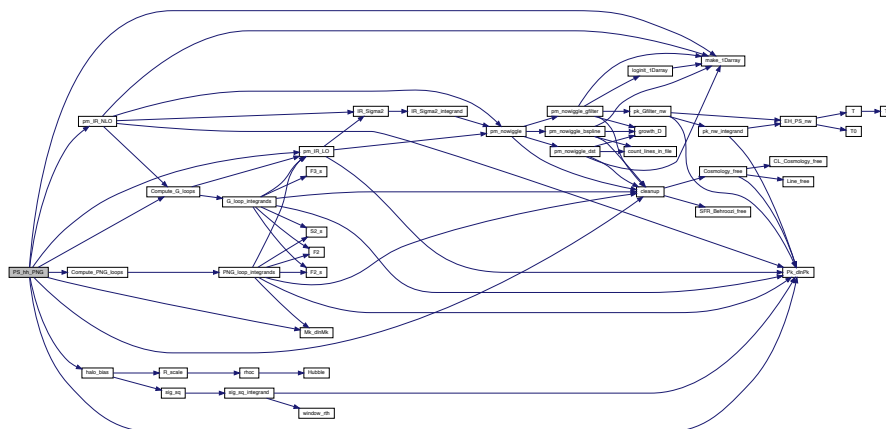
Parameters

<i>mode_mf</i>	Input \leftrightarrow : switch to set the theoretical model of the mass function used to compute the halo biases
----------------	---

Returns

PNG loop contributions of P_h

Here is the call graph for this function:



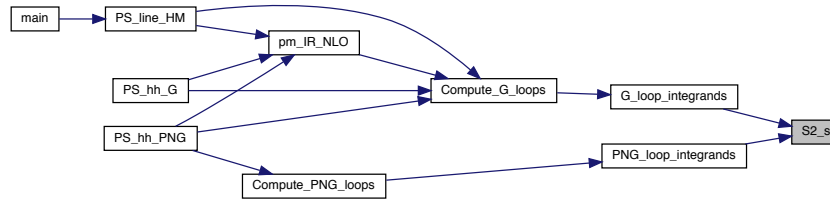
```
5.10.2.10 S2() double S2 (
                double mu )
```

```

5.10.2.11 S2_s() double S2_s (
    double k1,
    double k2,
    double mu )

```

Here is the caller graph for this function:



5.10.3 Variable Documentation

```

5.10.3.1 gb struct globals gb

```

5.11 ps_line_hm.c File Reference

Documented halo-model computation of line power spectrum, including clustering and stochastic contributions beyond Poisson limit.

```
#include "header.h"
```

Include dependency graph for ps_line_hm.c:



Functions

- double **PS_line_HM** (struct `Cosmology` *Cx, double k, double z, double M_min, long mode_mf, long line_type, int line_id)
Compute the clustering contribution to the line power spectrum using halo-model.
- double **PS_shot_HM** (struct `Cosmology` *Cx, double k, double z, double M_min, double *input, long mode_mf, long line_type)
Compute the shot noise contributions, including corrections beyond poisson limit (see 1706.08738 for more details) If nfw=1, the dependance of the power spectrum on the halo profile is neglected.
- static int **mhmc_integ** (const int *ndim, const cubareal x[], const int *ncomp, cubareal ff[], void *p)
Compute the corrections to mass integration of HM matter power spectrum.
- void **mhmc** (struct `Cosmology` *Cx, double z, long mode_mf, double *result)
- static int **HM_1h2h_integ** (const int *ndim, const cubareal x[], const int *ncomp, cubareal ff[], void *p)
Compute the mass integrals to compute the 1- and 1-halo line, line-matter and matter power spectrum If nfw=1, the dependance of the power spectrum on the halo profile is neglected.
- void **HM_1h2h** (struct `Cosmology` *Cx, double k, double z, double M_min, long mode_mf, long line_type, long mode_hm, double *result)
in unit of $M_{\text{sun}}/\text{Mpc}^3$
- static int **b22_Is_integrand** (const int *ndim, const cubareal x[], const int *ncomp, cubareal ff[], void *p)
Compute the large-scale limit of P_{b2b2} loop.
- double **b22_Is** (struct `Cosmology` *Cx, double z)

Variables

- struct globals gb

5.11.1 Detailed Description

Documented halo-model computation of line power spectrum, including clustering and stochastic contributions beyond Poisson limit.

Azadeh Moradinezhad Dizgah, November 4th 2021

This module has two main functions:

- `PS_line_HM()` to compute clustering (1- and 2-halo terms). The 2-halo term, includes nonlinear corrections to halo power spectrum arising from nonlinearities of matter fluctuations and halo biases.
- `PS_shot_HM()` to compute the stochastic contributions beyond Poisson shot noise (see arXiv:1706.08738)

The other functions in these modules are utilities for computing the above two main functions.

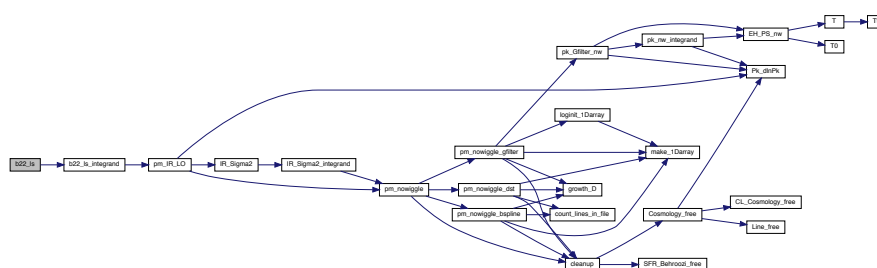
In summary, the following functions can be called from other modules:

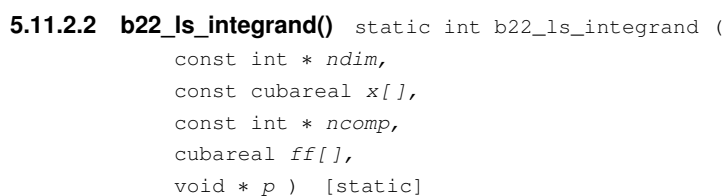
1. `PS_line_HM()`
2. `PS_shot_HM()`
3. `mhmc()` computes the corrections to mass integration of halo-model matter power spectrum
4. `HM_1h2h()` performs the mass integrals for computing 1- and 2-halo terms of line-line, line-matter and matter-matter power spectra.
5. `b22_ls()` computes the large-scale limit of P_{b2b2} loop which behaves like a constant and so contributes to the shot noise.

5.11.2 Function Documentation

```
5.11.2.1 b22_ls() double b22_ls (
    struct Cosmology * Cx,
    double z )
```

Here is the call graph for this function:





Compute the large-scale limit of P_b2b2 loop.

Cx	Input↔ : Cosmology struc- ture
z	Input↔ : red- shift

b22_ls

Here is the caller graph for this function:



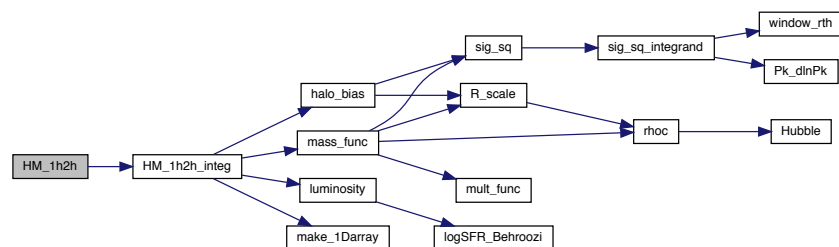
5.11.2.3 HM_1h2h() void HM_1h2h (

```

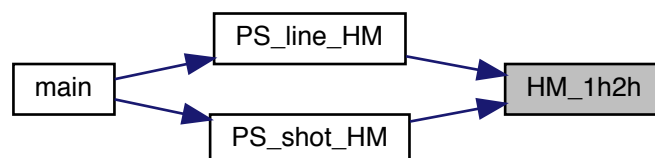
    struct Cosmology * Cx,
    double k,
    double z,
    double M_min,
    long mode_mf,
    long line_type,
    long mode_hm,
    double * result )
  
```

in unit of $M_{\text{sun}}/\text{Mpc}^3$

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.11.2.4 HM_1h2h_integ() static int HM_1h2h_integ (
    const int * ndim,
    const cubareal x[],
    const int * ncomp,
    cubareal ff[],
    void * p ) [static]

```

Compute the mass integrals to compute the 1- and 1-halo line, line-matter and matter power spectrum. If $n_{fw}=1$, the dependence of the power spectrum on the halo profile is neglected.

Otherwise, NFW halo profile is assumed

Parameters

C_x	Input↔ : Cosmology structure
k	Input↔ : wavenum- ber in unit of 1/Mpc.
z	Input↔ : red- shift
M_{min}	Input↔ : min- imum halo mass for mass inte- grals
$mode_{mf}$	Input↔ : theo- retical model of halo mass func- tion to use. It can be set to sheth- ↔ Tormen (ST), Tinker (TR) or Press- ↔ Schechter (PSC)

Parameters

<i>line_type</i>	Input↔ : name of the line to com- pute. It can be set to CII, CO10, CO21, CO32, CO43, CO54, CO65
------------------	--

Parameters

<i>mode_hm</i>	<p>Input↔</p> <p>: a switch to decide whether to compute the mass integrations. It can be set to:</p> <ul style="list-style-type: none">• LINE for line power spectrum,• LINEMATTER for line-matter cross spectrum• MATTER for matter power spectrum
----------------	--

Parameters

<i>results</i>	<p>Output : an array of the integration results. Number of elements varies depending on mode↵ _hm switch↵ :</p> <ul style="list-style-type: none">• 3 elements if mode↵ _↵ hm = LINE,• 1 element if mode↵ _↵ hm = LINE-MATTER• 2 element if mode↵ _↵ hm = MATTER results[0]↵
Generated by Doxygen	<p>: correction :</p>

Parameters

Returns

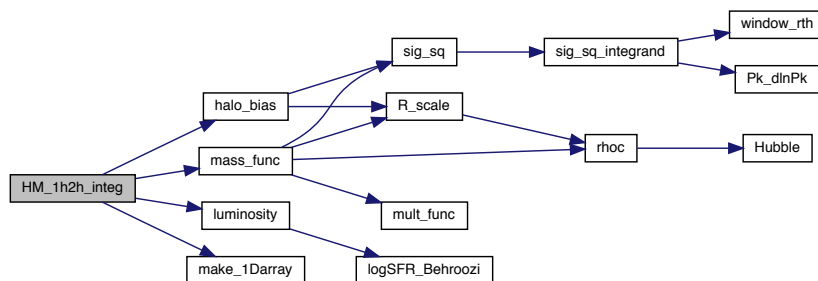
void

we assume the profile of both matter and line are NFW

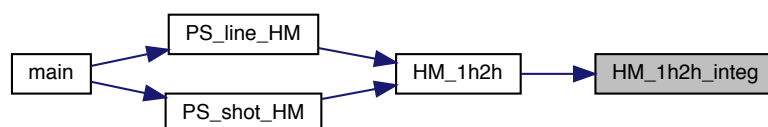
integrand of line 1halo term

integrand of 2halo term proportional to b_1 , the linear local-in-matter halo bias

integrand of 1halo term of line-matter cross-spectrum Here is the call graph for this function:



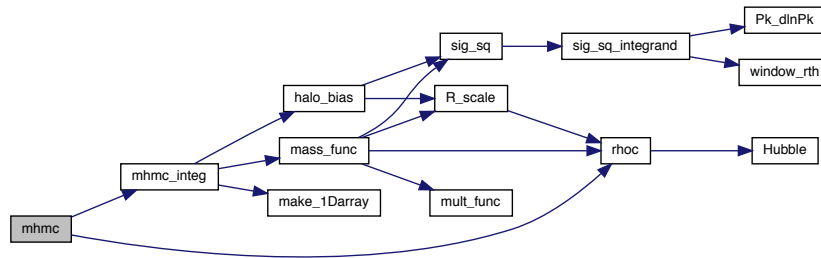
Here is the caller graph for this function:



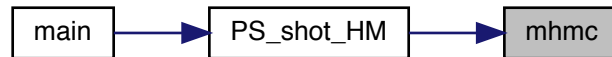
```

5.11.2.5 mhmc() void mhmc (
    struct Cosmology * Cx,
    double z,
    long mode_mf,
    double * result )
  
```


Here is the call graph for this function:



Here is the caller graph for this function:



```

5.11.2.6 mhmc_integ() static int mhmc_integ (
    const int * ndim,
    const cubareal x[],
    const int * ncomp,
    cubareal ff[],
    void * p ) [static]
  
```

Compute the corrections to mass integration of HM matter power spectrum.

The function [mhmc_integ\(\)](#) defines the integrand to be used by Cuhre integration routine of CUBA library. [mhmc\(\)](#) returns the corrections to 1- and 2-halo terms performing the integration

When computing the matter power spectrum using halo-model, the mass integrations for 1- and 2-loop terms get contributions from halos of all masses. For numerical computation, we need to impose a lower and upper integration limit. While the result of the integration are not sensitive to the upper bound (due to the fact that the mass function drops rapidly at high M_h) the choice of the lower bound affects the results. We can compute the leading order corrections to the integral that are accurate up to $(k R_s)^2$. (see App. A of arXiv:1511.02231 for more details.)

Parameters

Cx	Input↔ : Cosmology struc- ture
----	--

Parameters

z	Input↵ : red-shift
<i>mode_mf</i>	Input↵ : theoretical model of halo mass function to use. It can be set to Press-↵ Schecter (PSC), sheth-↵ Tormen (ST), Tinker (TR)

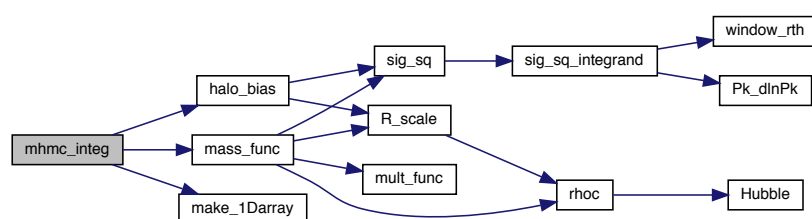
Parameters

<i>results</i>	<div>Output↵</div> <div>: a 2d array of the integration results,</div> <div><ul style="list-style-type: none">• <div>results[0]↵</div><div>:</div><div>correc-tion to 1-halo term,</div>• <div>result[1]↵</div><div>:</div><div>correcrions to 2-halo term as-sum-ing lin-ear halo bias</div></div>
----------------	--

Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.7 PS_line_HM() `double PS_line_HM (`
 `struct Cosmology * Cx,`
 `double k,`
 `double z,`
 `double M_min,`
 `long mode_mf,`
 `long line_type,`
 `int line_id)`

Compute the clustering contribution to the line power spectrum using halo-model.

If `nfw=1`, the dependance of the power spectrum on the halo profile is neglected. Otherwise, NFW halo profile is assumed

Parameters

<i>Cx</i>	Input↔ : pointer to <i>Cosmology</i> struc- ture
<i>k</i>	Input↔ : wavenum- ber in unit of 1/Mpc.
<i>z</i>	Input↔ : red- shift
<i>M_min</i>	Input↔ : min- imum halo mass for mass inte- grals

Parameters

<i>mode_mf</i>	Input↵ : theoretical model of halo mass function to use. It can be set to sheth-↵ Tormen (ST), Tinker (TR) or Press-↵ Schechter (PSC)
<i>line_type</i>	Input↵ : name of the line to compute. It can be set to CII, CO10, CO21, CO32, CO43, CO54, CO65
<i>line_id</i>	Input↵ : id of the line to be considered.

Returns

P_clust(k)

Boltzmann constant in unit of erg K⁻¹

in unit of erg/s

CII

Parameters

C_x	Input↔ : Cosmology struc- ture
k	Input↔ : wavenum- ber in unit of 1/Mpc.
z	Input↔ : red- shift
M_{min}	Input↔ : min- imum halo mass for mass inte- grals
<i>input</i>	input↔ : an array of input values with 4 values, Tave↔ _line, b1↔ _line, pb22↔ _ls, line↔ _shot, rhom↔ _bar

Parameters

<i>mode_mf</i>	Input↵ : theoretical model of halo mass function to use. It can be set to sheth-↵ Tormen (ST), Tinker (TR) or Press-↵ Schechter (PSC)
<i>line_type</i>	Input↵ : name of the line to compute. It can be set to CII, CO10, CO21, CO32, CO43, CO54, CO65

Returns

P_stoch(k)

Boltzmann constant in unit of erg K⁻¹

in unit of erg/s

CII

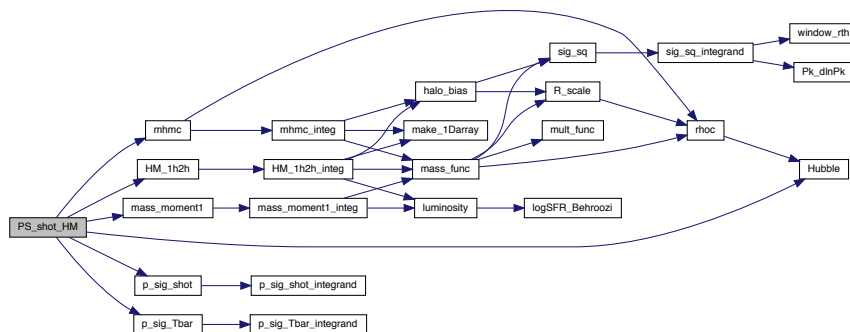
Since the following quantities do not depend on k, I am computing them once and pass them as input to this function

to plot the power spectrum in units of micro K² Mpc³

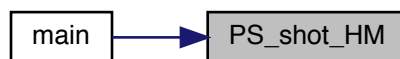
in unit of M_{sun}/Mpc³

in unit of M_{sun}/Mpc³

in unit of $M_{\text{sun}}/\text{Mpc}^3$ Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3 Variable Documentation

5.11.3.1 gb struct `globals` gb

5.12 ps_line_pt.c File Reference

Documented computation of Poisson shot noise and tree-level line power spectrum in real and redshift-space.

```
#include "header.h"
Include dependency graph for ps_line_pt.c:
```



Functions

- double `PS_line` (struct `Cosmology` *Cx, double k, double z, size_t line_id)
Compute the real-space 3D power spectrum of emission lines in unit of $\mu K^2 \text{ Mpc}^3$.
- double `PS_line_RSD` (struct `Cosmology` *Cx, struct `Cosmology` *Cx_ref, double k, double mu, double z, size_t line_id)
Compute the redshift-space 3D power spectrum of emission lines in unit of $\mu K^2 \text{ Mpc}^3$ as a function of wavenumber and angle w.r.t.
- int `ps_line_multipoles_integrand` (unsigned ndim, const double *x, void *p, unsigned fdim, double *fvalue)
Compute the multipole moments of redshift-space power spectrum of emission lines in unit of $\mu K^2 \text{ Mpc}^3$, integrating over the angle w.r.t LOS, weighted by.
- double `ps_line_multipoles` (struct `Cosmology` *Cx, struct `Cosmology` *Cx_ref, double k, double z, size_t line_id, int ell)
- double `PS_shot` (struct `Cosmology` *Cx, double z, size_t line_id)
Compute the Poisson shot noise in unit of $\mu K^2 \text{ Mpc}^3$.

Variables

- struct `globals gb`

5.12.1 Detailed Description

Documented computation of Poisson shot noise and tree-level line power spectrum in real and redshift-space.

Azadeh Moradinezhad Dizgah, November 4th 2021

NOTE TODO: Add the 1loop redshift-space power spectrum of the line. This requires implementing FFTLog, still in progress For the moment we stick to the tree-level expression of line power spectrum in redshift-space.

In summary, the following functions can be called from other modules:

1. `PS_line()` computes tree-level line power spectrum in real-space
2. `PS_line_RSD()` computes the tree-level line power spectrum in redshift-space, as a function of wavenumber and angle w.r.t LOS
3. `ps_line_multipoles()` computes the redshift-space multipoles of the line power spectrum
4. `PS_shot()` computes the poisson shot noise

5.12.2 Function Documentation

5.12.2.1 PS_line() double PS_line (
 struct `Cosmology` * Cx,
 double k,
 double z,
 size_t line_id)

Compute the real-space 3D power spectrum of emission lines in unit of $\mu K^2 \text{ Mpc}^3$.

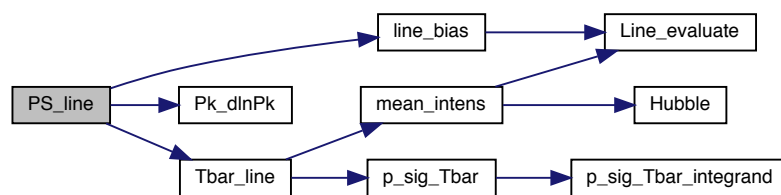
Parameters

Cx	Input↔ : Cosmology struc- ture
k	Input↔ : wavenum- ber in unit of 1/Mpc.
z	Input↔ : red- shift
<i>line_id</i>	Input↔ : id of the line to be con- sid- ered.

Returns

tree-level $P_{\text{clust}}(k)$

Here is the call graph for this function:

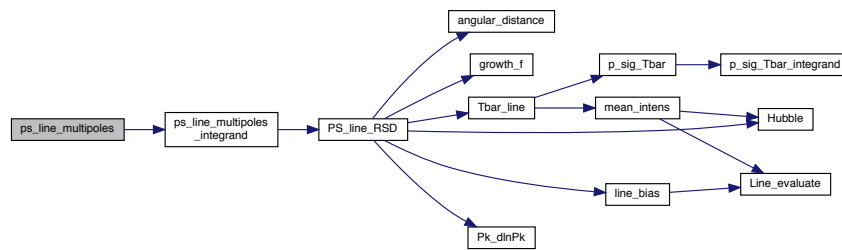


```

5.12.2.2 ps_line_multipoles() double ps_line_multipoles (
    struct Cosmology * Cx,
    struct Cosmology * Cx_ref,
    double k,
    double z,
    size_t line_id,
    int ell )

```

Here is the call graph for this function:



5.12.2.3 ps_line_multipoles_integrand() `int ps_line_multipoles_integrand (`
 unsigned *ndim*,
 const double * *x*,
 void * *p*,
 unsigned *fdim*,
 double * *fvalue*)

Compute the multipole moments of redshift-space power spectrum of emission lines in unit of $\mu\text{K}^2 \text{Mpc}^3$, integrating over the angle w.r.t LOS, weighted by.

Parameters

<i>Cx</i>	Input↔ : Cosmology structure
<i>Cx_ref</i>	Input↔ : Reference cos- mol- ogy struc- ture, needed for AP effect
<i>k</i>	Input↔ : wavenum- ber in unit of 1/Mpc.
<i>z</i>	Input↔ : red- shift

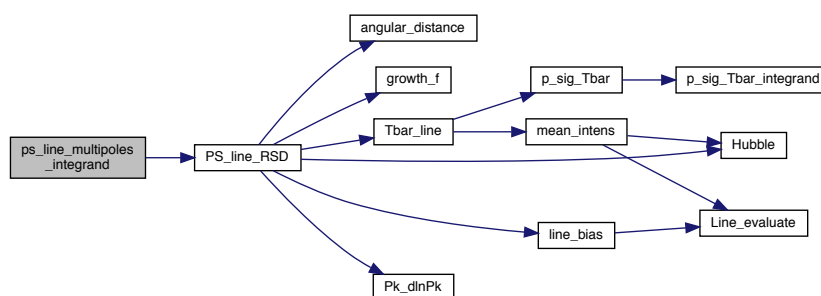
Parameters

<i>line_id</i>	Input↔ : id of the line to be con- sid- ered.
<i>ell</i>	Input↔ : the multi- pole

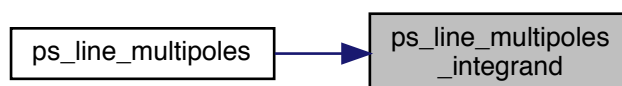
Returns

$P_{\text{ell}}(k)$

Here is the call graph for this function:



Here is the caller graph for this function:



```

5.12.2.4 PS_line_RSD() double PS_line_RSD (
    struct Cosmology * Cx,
    struct Cosmology * Cx_ref,
    double k,
    double mu,
    double z,
    size_t line_id )

```

Compute the redshift-space 3D power spectrum of emission lines in unit of $\mu\text{K}^2 \text{Mpc}^3$ as a function of wavenumber and angle w.r.t.

LOS

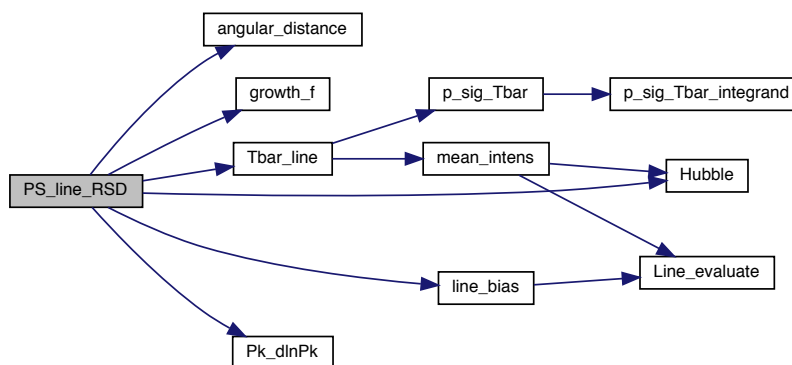
Parameters

<i>Cx</i>	Input↔ : <i>Cosmology</i> structure
<i>Cx_ref</i>	Input↔ : Reference cos- mol- ogy struc- ture, needed for AP effect
<i>k</i>	Input↔ : wavenum- ber in unit of 1/Mpc.
<i>mu</i>	Input↔ : angle w.r.t LOS
<i>z</i>	Input↔ : red- shift
<i>line_id</i>	Input↔ : id of the line to be con- sid- ered.

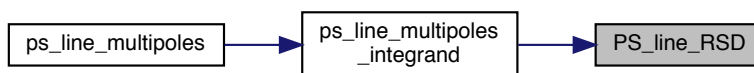
Returns

tree-level $P_{\text{clust}}(k, \mu)$

to plot the power spectrum in units of $\mu\text{K}^2 \text{Mpc}^3$. Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.5 PS_shot() double PS_shot (
 struct [Cosmology](#) * Cx,
 double z,
 size_t line_id)

Compute the Poisson shot noise in unit of $\mu\text{K}^2 \text{Mpc}^3$.

Parameters

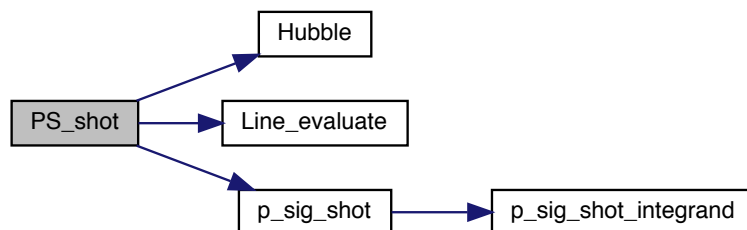
Cx	Input↔ : Cosmology struc- ture
z	Input↔ : red- shift
line_id	Input↔ : id of the line to be con- sid- ered

Returns

P_poisson
in unit of $\mu\text{K}^2 \text{Mpc}^3$

Boltzmann constant in unit of erg K^{-1}

in unit of erg/s Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3 Variable Documentation

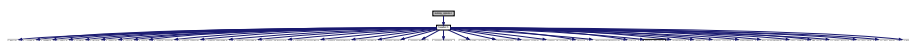
5.12.3.1 gb `struct globals gb`

5.13 survey_specs.c File Reference

Documented computation of some survey-related functions.

```
#include "header.h"
```

Include dependency graph for `survey_specs.c`:



Functions

- double `shell_volume` (struct `Cosmology` *Cx, double z, double fsky)
Compute the comoving volume of a survey covering redshift up to z.
- double `kmin_val` (struct `Cosmology` *Cx, double zmin, double zmax, double fsky)
Compute the size of fundamental mode corresponding to the comoving volume enclosed in a redshift bin [zmin,zmax].
- double `kmax_Brent` (double kmax, void *params)
Compute the maximum k-value used in Fisher forecast at each redshift bin.
- double `kmax_Brent_solver` (struct `Cosmology` *Cx, double z)

Variables

- struct `globals` `gb`

5.13.1 Detailed Description

Documented computation of some survey-related functions.

Azadeh Moradinezhad Dizgah, November 4th 2021

In summary, the following functions can be called from other modules:

1. `shell_volume()` computes the comoving volume of a survey covering redshift up to z
2. `kmin_val()` computes the fundamental k-mode of a given redshift shell
3. `kmax_Brent_solver()` computes the kmax value such that $k_{\max}(z=0) = 0.15 \text{ h/Mpc}$

5.13.2 Function Documentation

5.13.2.1 `kmax_Brent()` `double kmax_Brent (`
`double kmax,`
`void * params)`

Compute the maximum k-value used in Fisher forecast at each redshift bin.

We follow Giannantonio et al. to for determining kmax, and use gsl Brent solver to solve for kmax in each redshift bin. The goal is to compute the kmax such that at $z=0$, the variance of the matter fluctuations has a fixed value, for instance 0,36. This can be achieved by solving Eq. 40 of Giannantonio: $\sigma^2(z) = \int_{k_{\min}}^{k_{\max}(z)} dk \frac{k^2}{(2\pi)^2} P_m(k,z) = 0.36$. Instead of fixing σ^2 to 0.36, I chose the variance such that $k_{\max}(z=0) = 0.15 \text{ h/Mpc}$. This corresponds to the variance of ~ 0.33 at $z=0$. In the forecast, I additionally always impose $k_{\max} < 0.3 \text{ h/Mpc}$ cut.

Parameters

Cx	Input↔ : Cosmology struc- ture
z	Input↔ : red- shift of inter- est

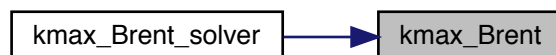
Returns

k_{\max}

Here is the call graph for this function:



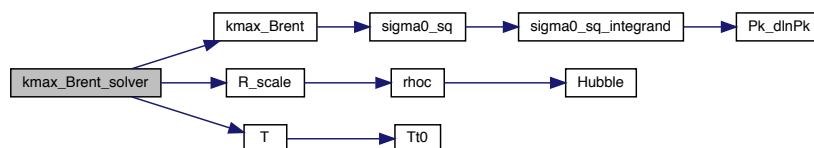
Here is the caller graph for this function:



5.13.2.2 kmax_Brent_solver() `double kmax_Brent_solver (`
`struct Cosmology * Cx,`
`double z)`

in short paper we used, 3.631872e-01 ;

Here is the call graph for this function:



5.13.2.3 kmin_val() `double kmin_val (`
 `struct Cosmology * Cx,`
 `double zmin,`
 `double zmax,`
 `double fsky)`

Compute the size of fundamental mode corresponding to the comoving volume enclosed in a redshift bin [zmin,zmax].

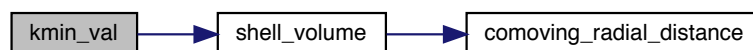
Parameters

<i>Cx</i>	Input↵ : <i>Cosmology</i> struc- ture
<i>zmin</i>	Input↵ : min- imum red- shift
<i>zmin</i>	Input↵ : max- imum red- shift
<i>fsky</i>	Input↵ : sky- coverage of teh survey

Returns

kmin

Here is the call graph for this function:



5.13.2.4 shell_volume() `double shell_volume (`
 `struct Cosmology * Cx,`
 `double z,`
 `double fsky)`

Compute the comoving volume of a survey covering redshift up to z .

Parameters

Cx	Input↔ : Cosmology struc- ture
z	Input↔ : red- shift
fsky	Input↔ : sky- coverage of teh survey

Returns

the comoving z-shell volume

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3 Variable Documentation

5.13.3.1 gb struct globals gb

5.14 utilities.c File Reference

Documented basic utility functions used by other modules of the code.

```
#include "header.h"
```

Include dependency graph for utilities.c:



Functions

- double * [make_1Darray](#) (long size)
Allocate memory to a 1d array of type double and length size.
- int * [make_1D_int_array](#) (long size)
Allocate memory to a 1d array of type integer and length size.
- double ** [make_2Darray](#) (long nrows, long ncolumns)
Allocate memory to a 2d array of type double.
- void [free_2Darray](#) (double **m)
Free the memory allocated to a 2d array.
- double * [init_1Darray](#) (long n, double xmin, double xmax)
initialize a 1d array, with values in the range of [xmin,xmax] and evenly-space on linear scale
- double * [loginit_1Darray](#) (long n, double xmin, double xmax)
initialize a 1d array, with values in the range of [xmin,xmax] and evenly-space on natural-log scale
- double * [log10init_1Darray](#) (long n, double inc, double xmin)
initialize a 1d array, with values in the range of [xmin,xmax] and evenly-space on log10 scale
- long [count_lines_in_file](#) (char *fname)
Count the number of lines of a file.
- long [count_cols_in_file](#) (char *fname)
Count the number of columns of a file.

5.14.1 Detailed Description

Documented basic utility functions used by other modules of the code.

Azadeh Moradinezhad Dizgah, November 4th 2021

In summary, the following functions can be called from other modules:

1. [make_1Darray\(\)](#) dynamically allocates memory to a 1d array
2. [make_2Darray\(\)](#) dynamically allocates memory to a 2d array
3. [free_2Darray\(\)](#) free the memory allocated to a 2d array
4. [init_1Darray\(\)](#) initialize a 1d array with linear spacing
5. [loginit_1Darray\(\)](#) initialize a 1d array with natural-log spacing
6. [log10init_1Darray\(\)](#) initialize a 1d array with log10 spacing
7. [count_lines_in_file\(\)](#) count the number of lines of a file
8. [count_cols_in_file\(\)](#) count number of columns of a file
9. [return_arr\(\)](#)

5.14.2 Function Documentation

5.14.2.1 count_cols_in_file() long count_cols_in_file (
char * fname)

Count the number of columns of a file.

Parameters

<i>fname</i>	Input↵ : file- name
--------------	---------------------------

Returns

long integer value of ncols

5.14.2.2 count_lines_in_file() long count_lines_in_file (
char * fname)

Count the number of lines of a file.

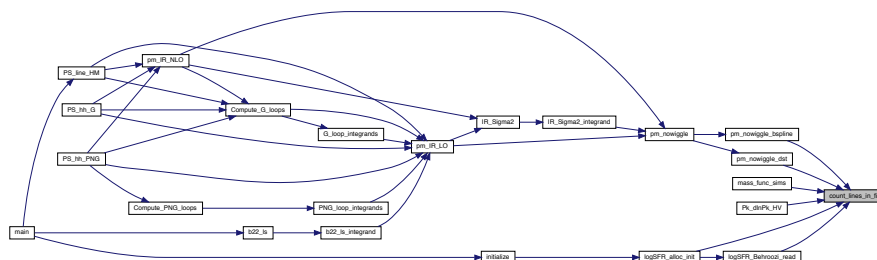
Parameters

<i>fname</i>	Input↵ : file- name
--------------	---------------------------

Returns

long integer value of nlines

Here is the caller graph for this function:



5.14.2.3 free_2Darray() `void free_2Darray (`
`double ** m)`

Free the memory allocated to a 2d array.

Parameters

<i>m</i>	Input↔ : double pointer to the elements of 2d array
----------	---

Returns

void

5.14.2.4 init_1Darray() `double * init_1Darray (`
`long n,`
`double xmin,`
`double xmax)`

initialize a 1d array, with values in the range of [xmin,xmax] and evenly-space on linear scale

Parameters

<i>n</i>	Input↔ : number of elements
<i>xmin</i>	Input↔ : start point
<i>xmi</i> ax	Input↔ : end point

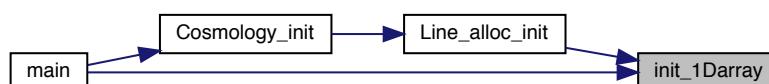
Returns

a pointer to a double type 1d array, with values initialized

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.5 log10init_1Darray() `double * log10init_1Darray (`
 `long n,`
 `double inc,`
 `double xmin)`

initialize a 1d array, with values in the range of [xmin,xmax] and evenly-space on log10 scale

Parameters

<i>n</i>	Input↔ : number of elements
<i>xmin</i>	Input↔ : start point
<i>xmiar</i>	Input↔ : end point

Returns

a pointer to a double type 1d array, with values initialized

Here is the call graph for this function:



5.14.2.6 loginit_1Darray() `double * loginit_1Darray (`
 `long n,`
 `double xmin,`
 `double xmax)`

initialize a 1d array, with values in the range of [*xmin*,*xmax*] and evenly-space on natural-log scale

Parameters

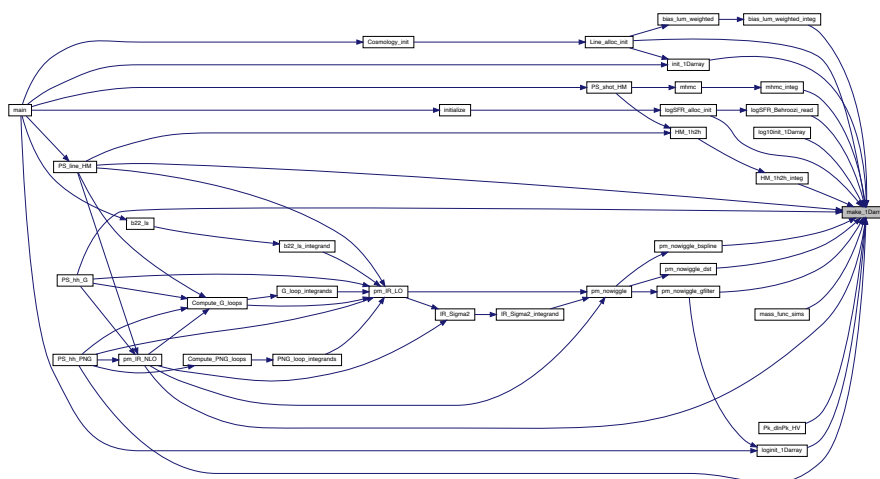
<i>n</i>	Input↵ : number of elements
<i>xmin</i>	Input↵ : start point
<i>xmi</i> <i>ax</i>	Input↵ : end point

Returns

a pointer to a double type 1d array, with values initialized

Here is the call graph for this function:





<i>nrows</i>	Input \leftarrow : number of rows of the output array
<i>ncols</i>	Input \leftarrow : number of columns of the output array

```
#include "header.h"
```

Include dependency graph for wnw_split.c:



Functions

- double [pk_Gfilter_nw](#) (struct [Cosmology](#) *Cx, double k, double k0)
Compute the nowiggle component of linear matter power spectrum using 3d Gaussian filter Computing the nowiggle component involves calculating an integral (Eq.
- double [pk_nw_integrand](#) (double x, void *par)
Integrand to compute the nowiggle matter power spectrum.
- double [EH_PS_w](#) (struct [Cosmology](#) *Cx, double k, double k0, double pk0)
Compute the Eisentein-Hu approximate wiggle component of linear matter power spectrum.
- double [EH_PS_nw](#) (struct [Cosmology](#) *Cx, double k, double k0, double pk0)
Compute the Eisentein-Hu approximate nowiggle component of linear matter power spectrum.
- double [T0](#) (struct [Cosmology](#) *Cx, double k)
Compute ????? AM:EDIT.
- double [T](#) (struct [Cosmology](#) *Cx, double k)
Compute the total baryon+CDM transfer function.
- double [Ti0](#) (struct [Cosmology](#) *Cx, double k, double x1, double x2)
Compute ????? AM:EDIT.

5.15.1 Detailed Description

Documented wiggle-nowiggle split based on 3d Gaussian filter in linear k, and using the Eisentein-Hu wiggle-no wiggle template

Azadeh Moradinezhad Dizgah, June 16th 2021

The algorithm closely follows Ref. arXiv:1509.02120 by Vlah et al. (described in Appendix A)

The following function will be called from other modules:

1. [pk_Gfilter_nw\(\)](#)

5.15.2 Function Documentation

5.15.2.1 EH_PS_nw() double EH_PS_nw (
 struct [Cosmology](#) * Cx,
 double k,
 double k0,
 double pk0)

Compute the Eisentein-Hu approximate nowiggle component of linear matter power spectrum.

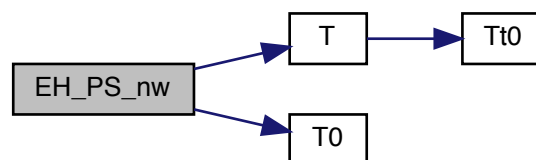
Parameters

Cx	Input↔ : pointer to Cosmology struc- ture
k	Input↔ : wavenum- ber in unit of 1/Mpc
$k0$	Input↔ : small- est value of k , i.e. the largest scale
$pk0$	Input↔ : value of the power spec- trum at the largest scale

Returns

$P_nw(k)$ in unit of $(\text{Mpc})^3$

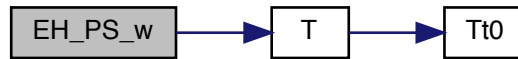
Here is the call graph for this function:



Returns

$P_w(k)$ in unit of $(\text{Mpc})^3$

Here is the call graph for this function:



```

5.15.2.3 pk_Gfilter_nw() double pk_Gfilter_nw (
    struct Cosmology * Cx,
    double k,
    double k0 )
  
```

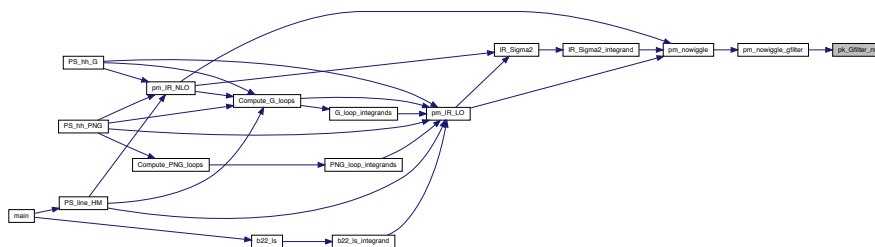
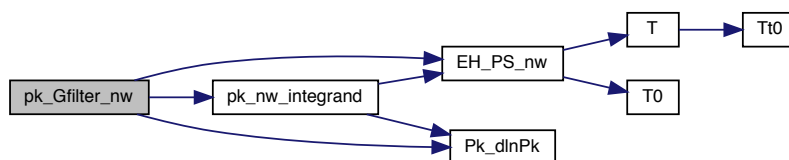
Compute the nowiggle component of linear matter power spectrum using 3d Gaussian filter Computing the nowiggle component involves calculating an integral (Eq.

A3 of Vlah et al) Below, [pk_nw_integrand\(\)](#) is the corresponding integrand and [pk_Gfilter_nw\(\)](#) is the integrator

Parameters

<i>Cx</i>	Input↔ : pointer to Cosmology struc- ture
<i>k</i>	Input↔ : wavenum- ber in unit of 1/Mpc
<i>k0</i>	Input↔ : small- est value of <i>k</i> , i.e. the largest scale

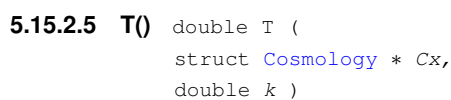
broadband component in unit of (Mpc)³



Integrand to compute the nowiggle matter power spectrum.

x	Input \leftarrow : inte- gration vari- able, k- values
par	Input \leftarrow : inte- gration param- eters

Here is the call graph for this function:



Parameters

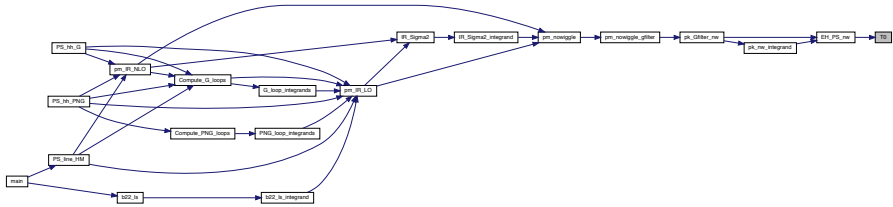
Returns

H0 value divided by the speed of light

Returns

value of ???

approximate sound speed given in Eq. (26) of EHHere is the caller graph for this function:



```
5.15.2.7 Tt0( double Tt0 (
    struct Cosmology * Cx,
    double k,
    double x1,
    double x2 )
```

Compute ????? AM:EDIT.

Parameters

Cx	Input↔ : pointer to Cosmology struc- ture
k	Input↔ : wavenum- ber in unit of 1/Mpc.
x1	Input↔ : betac AM↔ :WHAT WAS THIS VARI- ABLE???
x2	Input↔ : betac AM↔ :WHAT WAS THIS VARI- ABLE???

Index

- [_GNU_SOURCE](#)
 - [header.h, 22](#)
- [angular_distance](#)
 - [cosmology.c, 32](#)
- [As](#)
 - [globals, 8](#)
- [b1](#)
 - [globals, 8](#)
- [b1_LW_accel_ptr](#)
 - [Line, 16](#)
- [b1_LW_spline_ptr](#)
 - [Line, 17](#)
- [b22_ls](#)
 - [ps_line_hm.c, 127](#)
- [b22_ls_integrand](#)
 - [ps_line_hm.c, 128](#)
- [b2_LW_accel_ptr](#)
 - [Line, 17](#)
- [b2_LW_spline_ptr](#)
 - [Line, 17](#)
- [ba](#)
 - [Class_Cosmology_Struct, 5](#)
- [bias_lum_weighted](#)
 - [line_ingredients.c, 78](#)
- [bias_lum_weighted_integ](#)
 - [line_ingredients.c, 79](#)
- [BSPLINE](#)
 - [header.h, 22](#)
- [c](#)
 - [globals, 8](#)
- [ccs](#)
 - [Cosmology, 6](#)
- [CII](#)
 - [header.h, 22](#)
- [CL_Cosmology_free](#)
 - [cosmology.c, 33](#)
- [CL_Cosmology_initialize](#)
 - [cosmology.c, 34](#)
- [Class_Cosmology_Struct, 4](#)
 - [ba, 5](#)
 - [errmsg, 5](#)
 - [le, 5](#)
 - [nl, 5](#)
 - [op, 5](#)
 - [pm, 5](#)
 - [pr, 5](#)
 - [pt, 5](#)
 - [sp, 5](#)
 - [th, 5](#)
 - [tr, 5](#)
- [CLEANUP](#)
 - [header.h, 22](#)
- [cleanup](#)
 - [header.h, 26](#)
 - [main.c, 108](#)
- [CO10](#)
 - [header.h, 22](#)
- [CO21](#)
 - [header.h, 22](#)
- [CO32](#)
 - [header.h, 22](#)
- [CO43](#)
 - [header.h, 23](#)
- [CO54](#)
 - [header.h, 23](#)
- [CO65](#)
 - [header.h, 23](#)
- [comoving_radial_distance](#)
 - [cosmology.c, 35](#)
- [Compute_G_loops](#)
 - [ps_halo_1loop.c, 113](#)
- [Compute_PNG_loops](#)
 - [ps_halo_1loop.c, 115](#)
- [concentration_cdm](#)
 - [cosmology.c, 36](#)
- [cosmo_pars](#)
 - [Cosmology, 6](#)
- [Cosmology, 6](#)
 - [ccs, 6](#)
 - [cosmo_pars, 6](#)
 - [Lines, 6](#)
 - [mode_nu, 7](#)
 - [NLines, 7](#)
- [cosmology.c, 30](#)
 - [angular_distance, 32](#)
 - [CL_Cosmology_free, 33](#)
 - [CL_Cosmology_initialize, 34](#)
 - [comoving_radial_distance, 35](#)
 - [concentration_cdm, 36](#)
 - [Cosmology_free, 36](#)
 - [Cosmology_init, 37](#)
 - [der_Insig_sq, 40](#)
 - [derR_logwindow_g, 41](#)
 - [derR_window_rth, 42](#)
 - [gb, 61](#)
 - [growth_D, 42](#)
 - [growth_f, 43](#)
 - [Hubble, 44](#)
 - [Mk_dlnMk, 45](#)
 - [nfw_profile, 46](#)
 - [Pk_dlnPk, 47](#)
 - [Pk_dlnPk_HV, 49](#)
 - [R_scale, 51](#)
 - [R_vir, 52](#)
 - [rhoc, 53](#)
 - [sig_sq, 54](#)
 - [sig_sq_integrand, 56](#)
 - [sigma0_sq, 56](#)

- sigma0_sq_integrand, 57
 - window_g, 58
 - window_kth, 59
 - window_rth, 59
- Cosmology_free
 - cosmology.c, 36
- Cosmology_init
 - cosmology.c, 37
- count_cols_in_file
 - utilities.c, 157
- count_lines_in_file
 - utilities.c, 157
- data_dir
 - globals, 8
- data_priors
 - globals, 8
- DER
 - header.h, 23
- der_Insig_sq
 - cosmology.c, 40
- DERGROWTH
 - header.h, 23
- derNGLOOP
 - header.h, 23
- derR_logwindow_g
 - cosmology.c, 41
- derR_window_rth
 - cosmology.c, 42
- DO_NOT_EVALUATE
 - header.h, 23
- DST
 - header.h, 23
- EH_PS_nw
 - wnw_split.c, 163
- EH_PS_w
 - wnw_split.c, 165
- EQUILATERAL
 - header.h, 23
- errmsg
 - Class_Cosmology_Struct, 5
- F2
 - ps_halo_1loop.c, 117
- F2_s
 - ps_halo_1loop.c, 118
- F3_s
 - ps_halo_1loop.c, 118
- free_2Darray
 - utilities.c, 157
- G_loop_integrands
 - ps_halo_1loop.c, 118
- GAMMA
 - header.h, 23
- GAUSSIAN
 - header.h, 23
- gb
 - cosmology.c, 61
 - line_ingredients.c, 108
 - main.c, 111
 - ps_halo_1loop.c, 126
 - ps_line_hm.c, 143
 - ps_line_pt.c, 150
 - survey_specs.c, 155
- GFILTER
 - header.h, 24
- Global_Structs.h, 18
- globals, 7
 - As, 8
 - b1, 8
 - c, 8
 - data_dir, 8
 - data_priors, 8
 - h, 8
 - H0, 8
 - h_m, 8
 - kf, 8
 - kp, 8
 - logAs, 9
 - logM_accel_ptr, 9
 - logSFR_spline2d_ptr, 9
 - M_max, 9
 - M_min, 9
 - mass, 9
 - ng, 9
 - Npars, 9
 - ns, 9
 - Omega_b, 9
 - Omega_cdm, 9
 - Omega_g, 10
 - Omega_lambda, 10
 - Omega_nu, 10
 - Omega_r, 10
 - output_dir, 10
 - Planck_Fisher_filename, 10
 - project_home, 10
 - PS_kmax, 10
 - PS_kmin, 10
 - rho, 10
 - SFR_filename, 10
 - sigFOG0, 11
 - volume, 11
 - z_accel_ptr, 11
 - z_i, 11
 - z_max, 11
- GROWTH
 - header.h, 24
- growth_D
 - cosmology.c, 42
- growth_f
 - cosmology.c, 43
- h
 - globals, 8
- H0
 - globals, 8

h_m
 globals, 8
HALO
 header.h, 24
halo_bias
 line_ingredients.c, 80
header.h, 20, 28
 _GNU_SOURCE, 22
 BSPLINE, 22
 CII, 22
 CLEANUP, 22
 cleanup, 26
 CO10, 22
 CO21, 22
 CO32, 22
 CO43, 23
 CO54, 23
 CO65, 23
 DER, 23
 DERGROWTH, 23
 derNGLOOP, 23
 DO_NOT_EVALUATE, 23
 DST, 23
 EQUILATERAL, 23
 GAMMA, 23
 GAUSSIAN, 23
 GFILTER, 24
 GROWTH, 24
 HALO, 24
 HS, 24
 INIT, 24
 initialize, 26
 LINE, 24
 LINEAR, 24
 LINEMATTER, 24
 LOCAL, 24
 LOOP, 24
 LPOWER, 24
 MATTER, 25
 MAXL, 25
 NGLOOP, 25
 NLPOWER, 25
 NOIR, 25
 NONGAUSSIAN, 25
 NONLINEAR, 25
 ORTHOGONAL, 25
 PS_KMAX, 25
 PS_KMIN, 25
 PSC, 25
 QSF, 26
 QUADRATIC, 26
 ST, 26
 TIDE, 26
 TR, 26
 TRANS, 26
 TREE, 26
 WIR, 26
HM_1h2h
 ps_line_hm.c, 129
HM_1h2h_integ
 ps_line_hm.c, 129
HS
 header.h, 24
Hubble
 cosmology.c, 44

INIT
 header.h, 24
init_1Darray
 utilities.c, 158
initialize
 header.h, 26
 main.c, 109
initialized
 Line, 17
integrand_parameters, 11
 p1, 12
 p10, 12
 p11, 12
 p12, 12
 p13, 12
 p2, 12
 p3, 12
 p4, 12
 p5, 12
 p6, 12
 p7, 12
 p8, 13
 p9, 13
integrand_parameters2, 13
 p1, 14
 p10, 14
 p11, 14
 p12, 14
 p13, 14
 p14, 14
 p15, 14
 p16, 15
 p17, 15
 p18, 15
 p19, 15
 p2, 15
 p20, 15
 p22, 15
 p3, 15
 p4, 15
 p5, 15
 p6, 15
 p7, 16
 p8, 16
 p9, 16
IR_res.c, 61
 IR_Sigma2, 62
 IR_Sigma2_integrand, 64
 pm_IR_LO, 65
 pm_IR_NLO, 67
 pm_nowiggle, 68

- pm_nowiggle_bspline, 71
- pm_nowiggle_dst, 72
- pm_nowiggle_gfilter, 74
- IR_Sigma2
 - IR_res.c, 62
- IR_Sigma2_integrand
 - IR_res.c, 64
- kf
 - globals, 8
- kmax_Brent
 - survey_specs.c, 151
- kmax_Brent_solver
 - survey_specs.c, 152
- kmin_val
 - survey_specs.c, 153
- kp
 - globals, 8
- le
 - Class_Cosmology_Struct, 5
- LINE
 - header.h, 24
- Line, 16
 - b1_LW_accel_ptr, 16
 - b1_LW_spline_ptr, 17
 - b2_LW_accel_ptr, 17
 - b2_LW_spline_ptr, 17
 - initialized, 17
 - line_freq, 17
 - LineType, 17
 - mom1_accel_ptr, 17
 - mom1_spline_ptr, 17
 - mom2_accel_ptr, 17
 - mom2_spline_ptr, 17
 - npointsInterp, 17
- Line_alloc_init
 - line_ingredients.c, 82
- line_bias
 - line_ingredients.c, 84
- Line_evaluate
 - line_ingredients.c, 85
- Line_free
 - line_ingredients.c, 89
- line_freq
 - Line, 17
- line_ingredients.c, 76
 - bias_lum_weighted, 78
 - bias_lum_weighted_integ, 79
 - gb, 108
 - halo_bias, 80
 - Line_alloc_init, 82
 - line_bias, 84
 - Line_evaluate, 85
 - Line_free, 89
 - logSFR_alloc_init, 90
 - logSFR_Behroozi, 91
 - logSFR_Behroozi_read, 91
 - luminosity, 93
- mass_func, 94
- mass_func_sims, 95
- mass_moment1, 96
- mass_moment1_integ, 97
- mass_moment2, 99
- mass_moment2_integ, 100
- mean_intens, 101
- mult_func, 102
- p_sig_shot, 103
- p_sig_shot_integrand, 104
- p_sig_Tbar, 105
- p_sig_Tbar_integrand, 105
- SFR_Behroozi_free, 106
- Tbar_line, 106
- LINEAR
 - header.h, 24
- LINEMATTER
 - header.h, 24
- Lines
 - Cosmology, 6
- LineType
 - Line, 17
- LOCAL
 - header.h, 24
- log10init_1Darray
 - utilities.c, 159
- logAs
 - globals, 9
- loginit_1Darray
 - utilities.c, 160
- logM_accel_ptr
 - globals, 9
- logSFR_alloc_init
 - line_ingredients.c, 90
- logSFR_Behroozi
 - line_ingredients.c, 91
- logSFR_Behroozi_read
 - line_ingredients.c, 91
- logSFR_spline2d_ptr
 - globals, 9
- LOOP
 - header.h, 24
- LPOWER
 - header.h, 24
- luminosity
 - line_ingredients.c, 93
- M_max
 - globals, 9
- M_min
 - globals, 9
- main
 - main.c, 110
- main.c, 108
 - cleanup, 108
 - gb, 111
 - initialize, 109
 - main, 110
- make_1D_int_array

- utilities.c, [161](#)
- make_1Darray
 - utilities.c, [161](#)
- make_2Darray
 - utilities.c, [162](#)
- mass
 - globals, [9](#)
- mass_func
 - line_ingredients.c, [94](#)
- mass_func_sims
 - line_ingredients.c, [95](#)
- mass_moment1
 - line_ingredients.c, [96](#)
- mass_moment1_integ
 - line_ingredients.c, [97](#)
- mass_moment2
 - line_ingredients.c, [99](#)
- mass_moment2_integ
 - line_ingredients.c, [100](#)
- MATTER
 - header.h, [25](#)
- MAXL
 - header.h, [25](#)
- mean_intens
 - line_ingredients.c, [101](#)
- mhmc
 - ps_line_hm.c, [134](#)
- mhmc_integ
 - ps_line_hm.c, [135](#)
- Mk_dlnMk
 - cosmology.c, [45](#)
- mode_nu
 - Cosmology, [7](#)
- mom1_accel_ptr
 - Line, [17](#)
- mom1_spline_ptr
 - Line, [17](#)
- mom2_accel_ptr
 - Line, [17](#)
- mom2_spline_ptr
 - Line, [17](#)
- mult_func
 - line_ingredients.c, [102](#)
- nfw_profile
 - cosmology.c, [46](#)
- ng
 - globals, [9](#)
- NGLOOP
 - header.h, [25](#)
- nl
 - Class_Cosmology_Struct, [5](#)
- NLines
 - Cosmology, [7](#)
- NLPOWER
 - header.h, [25](#)
- NOIR
 - header.h, [25](#)
- NONGAUSSIAN
 - header.h, [25](#)
- NONLINEAR
 - header.h, [25](#)
- Npars
 - globals, [9](#)
- npointsInterp
 - Line, [17](#)
- ns
 - globals, [9](#)
- Omega_b
 - globals, [9](#)
- Omega_cdm
 - globals, [9](#)
- Omega_g
 - globals, [10](#)
- Omega_lambda
 - globals, [10](#)
- Omega_nu
 - globals, [10](#)
- Omega_r
 - globals, [10](#)
- op
 - Class_Cosmology_Struct, [5](#)
- ORTHOGONAL
 - header.h, [25](#)
- output_dir
 - globals, [10](#)
- p1
 - integrand_parameters, [12](#)
 - integrand_parameters2, [14](#)
- p10
 - integrand_parameters, [12](#)
 - integrand_parameters2, [14](#)
- p11
 - integrand_parameters, [12](#)
 - integrand_parameters2, [14](#)
- p12
 - integrand_parameters, [12](#)
 - integrand_parameters2, [14](#)
- p13
 - integrand_parameters, [12](#)
 - integrand_parameters2, [14](#)
- p14
 - integrand_parameters2, [14](#)
- p15
 - integrand_parameters2, [14](#)
- p16
 - integrand_parameters2, [15](#)
- p17
 - integrand_parameters2, [15](#)
- p18
 - integrand_parameters2, [15](#)
- p19
 - integrand_parameters2, [15](#)
- p2
 - integrand_parameters, [12](#)
 - integrand_parameters2, [15](#)

- p20
 - integrand_parameters2, [15](#)
- p22
 - integrand_parameters2, [15](#)
- p3
 - integrand_parameters, [12](#)
 - integrand_parameters2, [15](#)
- p4
 - integrand_parameters, [12](#)
 - integrand_parameters2, [15](#)
- p5
 - integrand_parameters, [12](#)
 - integrand_parameters2, [15](#)
- p6
 - integrand_parameters, [12](#)
 - integrand_parameters2, [15](#)
- p7
 - integrand_parameters, [12](#)
 - integrand_parameters2, [16](#)
- p8
 - integrand_parameters, [13](#)
 - integrand_parameters2, [16](#)
- p9
 - integrand_parameters, [13](#)
 - integrand_parameters2, [16](#)
- p_sig_shot
 - line_ingredients.c, [103](#)
- p_sig_shot_integrand
 - line_ingredients.c, [104](#)
- p_sig_Tbar
 - line_ingredients.c, [105](#)
- p_sig_Tbar_integrand
 - line_ingredients.c, [105](#)
- Pk_dlnPk
 - cosmology.c, [47](#)
- Pk_dlnPk_HV
 - cosmology.c, [49](#)
- pk_Gfilter_nw
 - wnw_split.c, [166](#)
- pk_nw_integrand
 - wnw_split.c, [167](#)
- Planck_Fisher_filename
 - globals, [10](#)
- pm
 - Class_Cosmology_Struct, [5](#)
- pm_IR_LO
 - IR_res.c, [65](#)
- pm_IR_NLO
 - IR_res.c, [67](#)
- pm_nowiggle
 - IR_res.c, [68](#)
- pm_nowiggle_bspline
 - IR_res.c, [71](#)
- pm_nowiggle_dst
 - IR_res.c, [72](#)
- pm_nowiggle_gfilter
 - IR_res.c, [74](#)
- PNG_loop_integrands
 - ps_halo_1loop.c, [119](#)
- pr
 - Class_Cosmology_Struct, [5](#)
- project_home
 - globals, [10](#)
- ps_halo_1loop.c, [111](#)
 - Compute_G_loops, [113](#)
 - Compute_PNG_loops, [115](#)
 - F2, [117](#)
 - F2_s, [118](#)
 - F3_s, [118](#)
 - G_loop_integrands, [118](#)
 - gb, [126](#)
 - PNG_loop_integrands, [119](#)
 - PS_hh_G, [120](#)
 - PS_hh_PNG, [123](#)
 - S2, [125](#)
 - S2_s, [125](#)
- PS_hh_G
 - ps_halo_1loop.c, [120](#)
- PS_hh_PNG
 - ps_halo_1loop.c, [123](#)
- PS_KMAX
 - header.h, [25](#)
- PS_kmax
 - globals, [10](#)
- PS_KMIN
 - header.h, [25](#)
- PS_kmin
 - globals, [10](#)
- PS_line
 - ps_line_pt.c, [144](#)
- PS_line_HM
 - ps_line_hm.c, [138](#)
- ps_line_hm.c, [126](#)
 - b22_ls, [127](#)
 - b22_ls_integrand, [128](#)
 - gb, [143](#)
 - HM_1h2h, [129](#)
 - HM_1h2h_integ, [129](#)
 - mhmc, [134](#)
 - mhmc_integ, [135](#)
 - PS_line_HM, [138](#)
 - PS_shot_HM, [140](#)
- ps_line_multipoles
 - ps_line_pt.c, [145](#)
- ps_line_multipoles_integrand
 - ps_line_pt.c, [146](#)
- ps_line_pt.c, [143](#)
 - gb, [150](#)
 - PS_line, [144](#)
 - ps_line_multipoles, [145](#)
 - ps_line_multipoles_integrand, [146](#)
 - PS_line_RSD, [147](#)
 - PS_shot, [149](#)
- PS_line_RSD
 - ps_line_pt.c, [147](#)
- PS_shot

- ps_line_pt.c, [149](#)
- PS_shot_HM
 - ps_line_hm.c, [140](#)
- PSC
 - header.h, [25](#)
- pt
 - Class_Cosmology_Struct, [5](#)
- QSF
 - header.h, [26](#)
- QUADRATIC
 - header.h, [26](#)
- R_scale
 - cosmology.c, [51](#)
- R_vir
 - cosmology.c, [52](#)
- README.md, [18](#)
- rho
 - globals, [10](#)
- rhoc
 - cosmology.c, [53](#)
- S2
 - ps_halo_1loop.c, [125](#)
- S2_s
 - ps_halo_1loop.c, [125](#)
- SFR_Behroozi_free
 - line_ingredients.c, [106](#)
- SFR_filename
 - globals, [10](#)
- shell_volume
 - survey_specs.c, [153](#)
- sig_sq
 - cosmology.c, [54](#)
- sig_sq_integrand
 - cosmology.c, [56](#)
- sigFOG0
 - globals, [11](#)
- sigma0_sq
 - cosmology.c, [56](#)
- sigma0_sq_integrand
 - cosmology.c, [57](#)
- sp
 - Class_Cosmology_Struct, [5](#)
- ST
 - header.h, [26](#)
- survey_specs.c, [150](#)
 - gb, [155](#)
 - kmax_Brent, [151](#)
 - kmax_Brent_solver, [152](#)
 - kmin_val, [153](#)
 - shell_volume, [153](#)
- T
 - wnw_split.c, [168](#)
- T0
 - wnw_split.c, [169](#)
- Tbar_line
 - line_ingredients.c, [106](#)
- th
 - Class_Cosmology_Struct, [5](#)
- TIDE
 - header.h, [26](#)
- TR
 - header.h, [26](#)
- tr
 - Class_Cosmology_Struct, [5](#)
- TRANS
 - header.h, [26](#)
- TREE
 - header.h, [26](#)
- Tt0
 - wnw_split.c, [170](#)
- utilities.c, [156](#)
 - count_cols_in_file, [157](#)
 - count_lines_in_file, [157](#)
 - free_2Darray, [157](#)
 - init_1Darray, [158](#)
 - log10init_1Darray, [159](#)
 - loginit_1Darray, [160](#)
 - make_1D_int_array, [161](#)
 - make_1Darray, [161](#)
 - make_2Darray, [162](#)
- volume
 - globals, [11](#)
- window_g
 - cosmology.c, [58](#)
- window_kth
 - cosmology.c, [59](#)
- window_rth
 - cosmology.c, [59](#)
- WIR
 - header.h, [26](#)
- wnw_split.c, [162](#)
 - EH_PS_nw, [163](#)
 - EH_PS_w, [165](#)
 - pk_Gfilter_nw, [166](#)
 - pk_nw_integrand, [167](#)
 - T, [168](#)
 - T0, [169](#)
 - Tt0, [170](#)
- z_accel_ptr
 - globals, [11](#)
- z_i
 - globals, [11](#)
- z_max
 - globals, [11](#)